UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Edwaldo Ramos de Brito Monteiro

MECANISMO DE PREVISÃO DE PERDA DE DEADLINE PARA A NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS

Florianópolis(SC)

2014

Edwaldo Ramos de Brito Monteiro

MECANISMO DE PREVISÃO DE PERDA DE DEADLINE PARA A NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS

Dissertação submetida ao Programa de Pós Graduação em Ciência da Computação como requisito para a obtenção do Grau de Mestre em Ciência da Computação.
Orientadora: Patricia Della Mea Plentz, Dra.Eng.

Florianópolis(SC)

2014

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Edwaldo Ramos de Brito Monteiro

MECANISMO DE PREVISÃO DE PERDA DE DEADLINE PARA A NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós Graduação em Ciência da Computação.

Florianópolis(SC), 1 de Outubro 2014.

Ronaldo dos Santos Mello
Coordenador

Banca Examinadora:

Patricia Della Mea Plentz, Dra.Eng.
Orientadora

Frank Augusto Siqueira, Prof. Dr.

Ricardo Alexandre Reinaldo De Moraes, Prof. Dr

Edson Roberto de Pieri, Prof. Dr

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer aos meus pais, Adriano de Brito Monteiro e Iolanda Vitorina Ramos Monteiro, e irmã, Elba Regina Ramos Monteiro, pelo imenso apoio durante toda minha vida acadêmica e investimento para que eu pudesse ter uma boa e sólida formação. Sem eles, nada disto seria possível!

Agradeço também à professora e minha orientadora, Patrícia Plentz pela paciência e prestabilidade demonstradas, pelas orientações, revisões e correções. Aos professores membros da banca avaliadora, Frank Siqueira, Ricardo Moraes e Edson de Pieri, por terem aceitado meu convite como membros da banca e auxiliado para a qualidade deste trabalho. Por fim, a todos os meus amigos, colegas e companheiros de curso, por me ajudarem nesta caminhada durante toda a minha pós graduação. Em especial, agradeço à Claudini Honório de Pieri e ao Tobias Monteiro por todo o apoio e companheirismo.

Aprendi através da experiência amarga a suprema lição: controlar minha ira e torná-la como o calor que é convertido em energia. Nossa ira controlada pode ser convertida numa força capaz de mover o mundo.

GANDHI

RESUMO

A robótica móvel vem ganhando um papel cada vez mais importante na sociedade moderna. Várias tarefas potencialmente perigosas ou penosas para o ser humano são atribuídas a robôs móveis, que são cada vez mais capazes. Muitas dessas tarefas precisam ser realizadas dentro de um prazo determinado, ou seja, devem cumprir um *deadline*. A perda desse *deadline* pode resultar em sérios prejuízos financeiros e/ou materiais. Assim sendo, mecanismos para prever a perda do *deadline* dessas tarefas são fundamentais, pois permitem que ações corretivas possam ser executadas no sentido de evitar ou minimizar os prejuízos decorrentes da perda do *deadline*. Neste trabalho, é proposto um mecanismo de previsão de perda de *deadline* confiável para robôs móveis autônomos. O mecanismo é capaz de prever a perda de *deadline* de uma determinada tarefa através da utilização de dados armazenados de execuções anteriores de tarefas similares. Foi utilizado o robô Pioneer 3-DX para experimentos e simulações, um dos robôs mais populares na academia. O mecanismo mostrou-se bastante eficaz e com alta taxa de previsão correta tanto em ambientes simulados quanto em ambientes reais.

Palavras-chave: Perda de deadline, robôs móveis, mecanismo de previsão, histórico de tarefas

ABSTRACT

Mobile robotics is gaining an increasingly important role in modern society. Several potentially dangerous or laborious tasks for human are assigned to mobile robots, which are increasingly capable. Many of these tasks need to be performed within a specified period, i.e, meet a deadline. Missing the deadline can result in financial and/or material losses. Therefore, mechanisms for predicting the missing of deadlines are fundamental because corrective actions can be taken to avoid or minimize the losses resulting from missing the deadline. In this work we propose a simple but reliable deadline missing prediction mechanism for mobile robots through the use of historical data and we use the Pioneer 3-DX robot for experiments and simulations, one of the most popular robots in academia. The mechanism proved to be very effective and with a high correct prediction rate, both in simulated environments and in real environments

Keywords: Deadline missing, mobile robots, prediction mechanism, historical data

LISTA DE FIGURAS

Figura 1 Modelo simplificado de um STR 29
Figura 2 O robô de exploração a Marte Curiosity 33
Figura 3 Ciclo de interação de um robô móvel com um ambiente 34
Figura 4 Modelo de tarefa em que o mecanismo pode ser aplicado 52
Figura 5 Arquitetura do sistema na visão de módulos 54
Figura 6 Diagrama de fluxo do processo de previsão 64
Figura 7 O robô Pionner 3-DX 66
Figura 8 O Laser Sick LMS-200 67
Figura 9 O laser LMS-200 acoplado ao Pioneer 3-DX 68
Figura 10 Interface gráfica do Mapper3 71
Figura 11 Interface gráfica do MobileEyes 72
Figura 12 Funcionamento do módulo de detecção implementado 73
Figura 13 Primeira configuração do ambiente 74
Figura 14 Segunda configuração do ambiente 74
Figura 15 Terceira configuração do ambiente 75
Figura 16 Simulação sendo executada no MobileSim 76
Figura 17 A taxa de previsões corretas nas simulações em ambientes simulados e estáticos 77
Figura 18 A influência do tamanho do histórico na taxa de previsões corretas 77
Figura 19 O mapa gerado do ambiente 79
Figura 20 Robô realizando tarefa 80
Figura 21 Robô evitando obstáculo 80
Figura 22 Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 1 81
Figura 23 Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 2 82
Figura 24 A influência de obstáculos fixos desconhecidos na taxa de previsões corretas no cenário 2, sem compensação 83
Figura 25 A influência de obstáculos móveis na taxa de previsões corretas no cenário 2, sem compensação 84
Figura 26 Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 3 86

Figura 27 A influência de obstáculos fixos desconhecidos na taxa de previsões corretas no cenário 3, com compensação 86
Figura 28 A influência de obstáculos móveis na taxa de previsões corretas no cenário 3, com compensação 87
Figura 29 A diferença entre as taxas de previsão correta obtidas nas simulações e nos testes experimentais 88
Figura 30 Comparação entre os mecanismos nas simulações em ambientes estáticos 90
Figura 31 Comparação entre os mecanismos nos testes em ambientes dinâmicos, sem mudança repentina no ambiente 92
Figura 32 Comparação entre os mecanismos nos testes em ambientes dinâmicos com mudanças frequentes 93

LISTA DE TABELAS

Tabela 1 Um exemplo do histórico principal do mecanismo 57
Tabela 2 Um exemplo do histórico de compensação 60
Tabela 3 Resultados da simulação. Tamanho do histórico = 1000 78
Tabela 4 Tabela comparativa entre mecanismos de previsão na Robótica Móvel 99

LISTA DE ABREVIATURAS E SIGLAS

STR	Sistemas de Tempo Real	29
RMS	Rate Monotonic Scheduling	31
EDF	Earliest Deadline First	31
LED	Light-Emitting Diode	36
CCD	Charged Coupled Device	37
GPS	Global Positioning System	37
TP	Tempo parcial de execução da tarefa	57
TT	Tempo total de execução da tarefa	57
NO	Número de obstáculos conhecidos no ambiente	57
Vel	Velocidade média do robô durante a execução da tarefa	57
VM	Velocidade máxima do robô durante a execução da tarefa	57
DL	Deadline da tarefa	58
TEP	Tempo de execução parcial	58
TMF	Tempo médio final	58
TMP	Tempo médio parcial	58
TDOF	Tempo desvio de obstáculos fixos	58
FDOM	Folga desvio de obstáculos móveis	58
TDOM	Tempo desvio de obstáculos móveis	58
TET	Tempo de execução total da tarefa	61
NPEC	Número de previsões erradas consecutivas	63
ARIA	Advanced Robotics Interface for Applications	68
API	Application Programming Interface	69

SUMÁRIO

1 INTRODUÇÃO 23
1.1 MOTIVAÇÃO E JUSTIFICATIVA 23
1.2 OBJETIVOS 26
1.3 ADEQUAÇÃO ÀS LINHAS DE PESQUISA DO CURSO 26
1.4 METODOLOGIA 26
1.5 ORGANIZAÇÃO DO TEXTO 28
2 CONCEITOS BÁSICOS 29
2.1 SISTEMAS DE TEMPO REAL 29
2.1.1 Escalonamento de tarefas 30
2.1.2 Criticalidade 31
2.1.3 Áreas de Aplicação 32
2.2 ROBÓTICA MÓVEL 33
2.2.1 Classificação dos robôs 34
2.2.2 Sensores 35
2.2.3 Atuadores 37
2.2.4 Arquitetura de robôs móveis 38
2.3 CONSIDERAÇÕES FINAIS 39
3 NAVEGAÇÃO DE ROBÔS MÓVEIS 41
3.1 LOCALIZAÇÃO DE ROBÔS MÓVEIS 41
3.1.1 Localização relativa 41
3.1.2 Localização absoluta 42
3.2 MODELAGEM DO AMBIENTE 44
3.3 TÉCNICAS DE NAVEGAÇÃO 44
3.3.1 Abordagem reativa 45
3.3.2 Abordagem representativa 45
3.4 TÉCNICAS DE PLANEJAMENTO DE TRAJETÓRIA 46
3.4.1 Roadmaps 47
3.4.2 Decomposição em Células 48
3.4.3 Campos potenciais 48
3.5 CONSIDERAÇÕES FINAIS 49
4 PROPOSTA 51
4.1 DESCRIÇÃO INICIAL 51
4.2 ARQUITETURA DO SISTEMA 52
4.3 O MECANISMO DE PREVISÃO 55
4.3.1 Histórico principal do mecanismo 56
4.3.2 Fórmula de previsão 57
4.3.3 Compensação de mudanças repentinas no ambiente 60

4.4 CONSIDERAÇÕES FINAIS 63
5 SIMULAÇÕES, TESTES E RESULTADOS 65
5.1 HARDWARE UTILIZADO 65
5.1.1 Robô Pioneer 3-DX 66
5.1.2 Laser Sick LMS-200 67
5.2 SOFTWARE UTILIZADO 68
5.2.1 ARIA 68
5.2.2 ARNL 69
5.2.3 MobileSim 70
5.2.4 Mapper3 70
5.2.5 MobileEyes 70
5.3 MÓDULO DE DETECÇÃO DE OBSTÁCULOS 71
5.4 SIMULAÇÕES E TESTES REALIZADOS 73
5.4.1 Ambiente simulado 73
5.4.1.1 Resultados 75
5.4.2 Ambiente real 78
5.4.2.1 Cenário 1 79
5.4.2.1.1 Resultados 80
5.4.2.2 Cenário 2 80
5.4.2.2.1 Resultados 82
5.4.2.3 Cenário 3 84
5.4.2.3.1 Resultados 85
5.4.3 Comparação entre simulação e avaliação experimental 87
5.5 VALIDAÇÃO DO MECANISMO 88
5.5.1 Mecanismo para comparação 89
5.5.2 Comparação entre os mecanismos 89
5.5.2.1 Cenário 1 89
5.5.2.1.1 Resultados 90
5.5.2.2 Cenário 2 91
5.5.2.2.1 Resultados 91
5.5.2.3 Cenário 3 91
5.5.2.3.1 Resultados 92
5.6 CONSIDERAÇÕES FINAIS 92
6 TRABALHOS RELACIONADOS 95
6.1 MECANISMOS DE PREVISÃO NA ROBÓTICA MÓVEL 95
6.2 MECANISMOS EM OUTRAS CLASSES DE SISTEMAS 97
6.3 CONSIDERAÇÕES FINAIS 100
7 CONSIDERAÇÕES FINAIS 101
7.1 CONCLUSÃO 101
7.2 TRABALHOS FUTUROS 102
Referências Bibliográficas 105

1 INTRODUÇÃO

O presente trabalho visa desenvolver um mecanismo de previsão de cumprimento de tarefa para robôs móveis autônomos durante a navegação, respeitando um tempo máximo de execução - *deadline*. Este capítulo apresenta uma introdução do que é elaborado neste trabalho, introduzindo ao leitor o contexto do mesmo, a motivação para a sua elaboração, a metodologia, bem como os objetivos a serem alcançados.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

Na sociedade atual existe uma necessidade cada vez maior de realizar tarefas com muita eficiência e precisão. Muitas dessas tarefas necessitam ser realizadas em locais arriscados e de difícil ou impossível acesso humano, como por exemplo, o fundo do mar ou o espaço. Essa necessidade vem impulsionando a criação de dispositivos, chamados robôs, que de forma praticamente autônoma e inteligente realizam essas tarefas sem risco para a vida humana. Um robô é um agente eletromecânico capaz de realizar diversas tarefas com assistência humana ou de forma totalmente independente. Nos últimos anos, tem-se notado um grande crescimento do potencial desses robôs, ganhando um papel cada vez mais importante no nosso cotidiano.

A robótica móvel tem sido foco de muito estudo e se tornou uma área de grande importância e interesse para pesquisadores e engenheiros. Robôs móveis são robôs que possuem a capacidade de se moverem em um determinado ambiente, não estando limitados a uma localização fixa como os robôs manipuladores industriais. Robôs móveis autônomos são capazes de realizar tarefas sem nenhuma assistência ou interação humana e possuem diversas aplicações no meio militar, indústria, medicina, entretenimento, pesquisa, etc. Um dos principais aspectos a ser considerado na robótica móvel é o ambiente onde o robô vai se locomover. O ambiente pode ser totalmente conhecido, parcialmente conhecido ou desconhecido. No primeiro, o ambiente é estático, isto é, nunca sofre modificações e o robô possui uma representação completa (um mapa) deste ambiente. Em um ambiente parcialmente conhecido, o robô também possui um mapa descrevendo suas informações, porém o ambiente pode sofrer modificações durante a execução de tarefas. Por fim, em um ambiente desconhecido o robô não possui nenhuma representação deste ambiente e a navegação é possível através da coleta de dados pelos sensores do robô. Para navegar num ambiente parcialmente conhecido, o robô geralmente faz uso de um mapa deste ambiente, que pode conter objetos desconhecidos,

áreas proibidas e obstáculos móveis. O robô utiliza as informações do mapa, juntamente com os dados obtidos pelos seus sensores, para navegar de forma segura. Outro aspecto importante é o planejamento de rota ou trajetória, que permite que uma rota segura seja definida da posição de origem do robô até o destino final, evitando obstáculos conhecidos e desconhecidos.

Sistemas de Tempo Real são sistemas computacionais em que o tempo de execução de uma determinada tarefa deve ser considerado. Estes sistemas, além de demandarem resultados logicamente corretos, possuem a característica de que suas respostas ao ambiente devem ser dadas em um tempo hábil para que o sistema não entre em um estado inconsistente ou inválido. O sistema deve ser capaz de executar a tarefa em um tempo limite ou informar que a mesma não poderá ser executada no prazo estabelecido (FARINES; FRAGA; OLIVEIRA, 2000). Neste sentido, a corretude do sistema depende não só dos resultados da computação, mas também do instante de tempo em que estes são produzidos. Muitas vezes, os Sistemas de Tempo Real são associados de forma errada a sistemas que produzem resultados rápidos. Porém, o que caracteriza um sistema como sendo de tempo real é essencialmente a restrição de que ele deve cumprir metas temporais explícitas (*deadlines*). Segundo Kopetz (2011), os Sistemas de Tempo Real podem ser classificados como críticos ou não críticos dependendo da severidade da pena pelo não cumprimento de uma tarefa dentro do intervalo de tempo estipulado. Nos sistemas críticos, o prazo para execução de uma tarefa não pode ser violado e a perda de um *deadline* pode implicar em consequências irreversíveis. Já nos sistemas não-críticos, o tempo continua sendo um parâmetro fundamental de execução, mas a perda de alguns *deadlines* é tolerável.

Os robôs móveis são um exemplo de Sistema de Tempo Real. São formados por um conjunto de subsistemas - subsistema de atuadores, subsistema de sensores e um subsistema de software, os quais trabalham em conjunto para que o robô respeite as restrições temporais das tarefas de baixo nível, como por exemplo, leitura de sensores e acionamento do motor do robô. É necessário que esses subsistemas funcionem corretamente para que o robô consiga realizar tarefas de alto nível, como por exemplo, navegar de uma posição inicial até uma posição final. Ignorando possíveis falhas nos subsistemas citados anteriormente e considerando que os mesmos realizam as tarefas de baixo nível no tempo estipulado, podemos aplicar restrições temporais também nas tarefas de alto nível do robô. Tais tarefas, além de logicamente corretas, precisam respeitar um determinado *deadline*. A possibilidade de prever a perda deste *deadline* é importante em diversos cenários. Algumas tarefas realizadas por robôs móveis autônomos devem ser concluídas dentro de um prazo máximo estabelecido e perdem a validade ou trazem consequências indesejáveis caso o prazo não seja cumprido. Por exemplo, numa

plataforma de petróleo automatizada, a não realização de uma determinada tarefa no tempo estipulado previamente pode causar atrasos e sérios prejuízos materiais e financeiros. No cenário de um depósito ou armazém, a tarefa de transporte de cargas pode ser atribuída a robôs móveis autônomos. Geralmente nestes ambientes existem prazos definidos para as atividades de carga e descarga de mercadorias, de modo a não comprometer a logística da empresa responsável e evitar possíveis prejuízos devido a atrasos no processo de transporte. Um robô móvel que receba a tarefa de transportar um determinado número de caixas do local de armazenamento até uma outra posição dentro de um depósito, deve concluir essa tarefa dentro de um determinado tempo sob pena de consequências indesejáveis para a empresa, como por exemplo, a perda do prazo de entrega das mercadorias ao cliente. Mecanismos capazes de prever se essa tarefa será ou não realizada no prazo estipulado são importantes pois permitem que ações corretivas possam ser executadas com o objetivo de evitar tais consequências. Neste exemplo, caso o mecanismo indique a perda do *deadline* da tarefa, as ações corretivas que podem ser realizadas são: seleção de um outro robô para ajudar no transporte das caixas, aumento da velocidade do robô para diminuir o tempo do transporte ou ainda o disparo de um alerta para que um operador humano tome a melhor decisão possível. A mesma ideia se aplica a diversos outros cenários onde a perda do *deadline* de uma tarefa de navegação do robô possa implicar em consequências indesejáveis ou até mesmo prejuízos financeiros e/ou materiais. Neste sentido, torna-se necessário desenvolver mecanismos confiáveis de previsão de perda de *deadline* na realização das tarefas do robô, de modo que ações corretivas possam ser executadas a tempo de evitar ou minimizar possíveis prejuízos.

Este trabalho visa desenvolver um mecanismo de previsão de perda de *deadline* para as tarefas de navegação dos robôs móveis autônomos, através do uso de dados armazenados de tarefas similares executadas anteriormente. Estas tarefas consistem em mover o robô de uma posição inicial até uma posição destino e na volta para a posição inicial. A execução destas tarefas caracteriza o transporte de cargas, prestação de assistência a outros robôs ou execução de outras funções dentro de um ambiente parcialmente conhecido e previsível, como por exemplo, um armazém, uma fábrica ou um depósito. Nestes ambientes, o robô pode encontrar obstáculos desconhecidos fixos e móveis durante a sua navegação, porém tais obstáculos são quase sempre previsíveis, como pessoas, caixas, outros robôs, etc.

Para a implementação e validação do mecanismo proposto, é utilizado o robô *Pioneer 3-DX*, um robô móvel muito conhecido e estudado no meio acadêmico.

1.2 OBJETIVOS

O objetivo geral desta dissertação é elaborar, desenvolver e avaliar um mecanismo de previsão de perda de *deadlines* para robôs móveis autônomos, na realização de tarefas de navegação dentro de ambientes parcialmente conhecidos. Para atender o objetivo geral desta dissertação, os seguintes objetivos específicos foram definidos:

- Definir um conjunto de tarefas similares em que o mecanismo de previsão possa ser aplicado;
- Elaborar um mecanismo simples mas eficiente de previsão de cumprimento de tarefa de robôs móveis autônomos respeitando um tempo máximo estipulado;
- Definir métricas para avaliação do mecanismo proposto;
- Avaliar a qualidade das previsões realizadas pelo mecanismo.

1.3 ADEQUAÇÃO ÀS LINHAS DE PESQUISA DO CURSO

O trabalho descrito nesta dissertação está inserido no contexto da área de Inteligência Computacional do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina. Este trabalho está perfeitamente integrado com os demais trabalhos de pesquisa sobre robótica móvel e com as atividades do Curso de Pós-Graduação em Ciência da Computação desta Universidade.

1.4 METODOLOGIA

Uma pesquisa pode ser classificada quanto à natureza, a forma de abordagem, aos objetivos e aos procedimentos técnicos.

Quanto à natureza, o presente trabalho se classifica como uma pesquisa aplicada, já que de acordo com Silva e Menezes (2005), a pesquisa aplicada tem por objetivo gerar conhecimentos para aplicação prática dirigida à solução de problemas específicos.

Quanto à forma de abordagem, trata-se de uma pesquisa qualitativa. Em Silva e Menezes (2005), a pesquisa qualitativa é descrita como

> Pesquisa Qualitativa: considera que há uma relação dinâmica entre o mundo real e o sujeito, isto é, um

> vínculo indissociável entre o mundo objetivo e a subjetividade do sujeito que não pode ser traduzido em números. A interpretação dos fenômenos e a atribuição de significados são básicos no processo de pesquisa qualitativa. Não requer o uso de métodos e técnicas estatísticas. O ambiente natural é a fonte direta para coleta de dados e o pesquisador é o instrumento chave. É descritiva. Os pesquisadores tendem a analisar seus dados indutivamente. O processo e seu significado são os focos principais de abordagem.

Quanto aos objetivos, pode-se classificar como pesquisa exploratória, que visa proporcionar maior familiaridade com o problema com vistas a torná-lo explícito ou a construir hipóteses. Assume, em geral, as formas de Pesquisas Bibliográficas e Estudos de caso (SILVA; MENEZES, 2005).

Quanto aos procedimentos técnicos, trata-se de um trabalho de estudo de caso e pesquisa bibliográfica. Ainda de acordo com Silva e Menezes (2005), estudo de caso é quando envolve o estudo profundo e exaustivo de um ou poucos objetos de maneira que permita o seu amplo e detalhado conhecimento e pesquisa bibliográfica quando é elaborada a partir de material já publicado, constituído principalmente de livros, artigos de periódicos e atualmente com material disponibilizado na Internet.

Na etapa inicial do projeto foi realizada a revisão bibliográfica. A primeira parte da pesquisa foi embasada em artigos, livros e documentos publicados na internet que abordam tempo real e robótica móvel. Em seguida foi feito um estudo dos trabalhos relacionados e mecanismos de previsão de perda de *deadline* já propostos anteriormente, nas diversas áreas da Computação.

A próxima etapa consistiu em estudar os hardwares e softwares que seriam utilizados no desenvolvimento e validação do mecanismo. Nesta etapa houve um primeiro contato com o robô Pioneer 3-DX e a realização de experimentos iniciais simples, mas que se mostraram muito úteis para o posterior projeto do mecanismo.

Na etapa seguinte, foi feito o projeto e implementação do mecanismo. Inicialmente, foi desenvolvido um mecanismo simples para ambientes simulados e estáticos. Após simulações com resultados satisfatórios nesses ambientes, foi feita uma extensão do projeto do mecanismo para ambientes dinâmicos.

Por fim, foi feita a validação do mecanismo com experimentos em ambientes reais.

1.5 ORGANIZAÇÃO DO TEXTO

O primeiro capítulo desta dissertação apresentou os aspectos que motivaram esta pesquisa, os objetivos deste trabalho, a metodologia utilizada e a adequação às linhas de pesquisa do curso.

No capítulo 2 são descritos conceitos sobre sistemas de tempo real e robótica móvel, importantes para a compreensão do restante do trabalho.

No capítulo 3 é feita uma revisão da literatura sobre técnicas de localização de um robô móvel, formas de representação de um ambiente, planejamento de trajetória e técnicas de navegação.

No capítulo 4 a proposta desta dissertação é abordada em maiores detalhes, apresentando a arquitetura e o funcionamento do mecanismo de previsão proposto.

O capítulo 5 faz a validação do mecanismo, mostrando as simulações e experimentos realizados, bem como os softwares e hardwares utilizados no projeto. Também são apresentados os resultados obtidos.

O capítulo 6 apresenta alguns trabalhos relacionados e o capítulo 7 apresenta as considerações finais deste trabalho e sugere alguns trabalhos futuros.

2 CONCEITOS BÁSICOS

Neste capítulo, são descritos alguns conceitos sobre sistemas de tempo real e robótica móvel que dão o embasamento teórico necessário para o entendimento do trabalho desenvolvido.

Estas áreas de pesquisa apresentam extensa teoria, sendo que muitos conceitos fogem ao escopo deste trabalho. Este capítulo fica, portanto, restrito à descrição dos conceitos necessários para o entendimento desta dissertação.

2.1 SISTEMAS DE TEMPO REAL

De acordo com Farines, Fraga e Oliveira (2000), Sistemas de Tempo Real (STR) são sistemas computacionais que devem reagir a estímulos oriundos do ambiente respeitando restrições temporais, ou seja, as suas tarefas devem produzir resultados logicamente corretos no tempo especificado. É definido um tempo máximo de execução para as tarefas e diferentes políticas de escalonamento podem ser aplicadas para garantir que as mesmas sejam executadas dentro do prazo estipulado. Deste modo, o correto comportamento de um sistema de tempo real não depende apenas da integridade dos resultados obtidos mas também do tempo em que são produzidos. O resultado de uma tarefa que ocorra além do prazo esperado, caracteriza uma falha do sistema e pode representar uma ameaça.

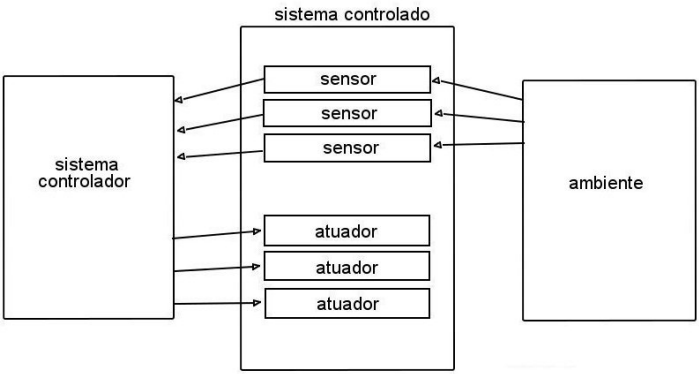

Figura 1: Modelo simplificado de um STR

De forma simplificada, um Sistema de Tempo Real é composto por um sistema controlador, um sistema controlado e um ambiente conforme mostra a Figura 1. O sistema controlador obtém informações sobre o ambiente através de sensores e controla o mesmo utilizando atuadores (KOPETZ, 2011). Um sensor é um dispositivo que responde a um estímulo físico e/ou químico de forma específica e mensurável enquanto que um atuador é um elemento que produz movimento, obedecendo comandos que podem ser manuais, elétricos ou mecânicos. Um exemplo seria um sistema que controla as pernas de um robô, que responde continuamente às mudanças do ambiente físico num determinado intervalo de tempo. Se ocorrer uma perda de *deadline* no controle das pernas, o robô pode cair.

Em sistemas onde as noções de tempo e de concorrência são tratadas explicitamente, conceitos e técnicas de escalonamento formam o ponto central na previsibilidade do comportamento dos Sistemas de Tempo Real. Na próxima seção, alguns conceitos relacionados ao escalonamento e criticalidade de STR são apresentados.

2.1.1 Escalonamento de tarefas

O sistema de software de um STR inclui o sistema operacional (ou *kernel*) e as tarefas da aplicação. O sistema operacional de tempo real é responsável pelo gerenciamento dos recursos do sistema, com o objetivo de garantir que todas as tarefas (ou o maior número possível) sejam realizadas dentro das suas restrições de tempo (LAPLANTE, 2004). Segundo Mall (2009), as tarefas são atividades que são executadas repetidamente e podem ser classificadas conforme a periodicidade das suas ativações. Uma tarefa é periódica se a cada período P de tempo sempre ocorrer uma ativação e aperiódica quando suas ativações são aleatórias, respondendo a eventos internos ou externos. Caso exista um intervalo mínimo de tempo (maior do que zero) entre duas ativações de uma tarefa aperiódica, ela é classificada como esporádica.

Assim como qualquer sistema operacional, um sistema operacional de tempo real utiliza uma fila de tarefas que estão prontas para serem executadas. Esta fila é conhecida como fila de prontos. Diferentes algoritmos de escalonamento podem ser utilizados para ordenar a fila de prontos visando satisfazer os requisitos temporais das tarefas. De acordo com Farines, Fraga e Oliveira (2000), esses algoritmos de escalonamento podem ser classificados como estáticos ou dinâmicos:

- os algoritmos estáticos são utilizados apenas em situações onde a tarefa a ser realizada e seus requisitos temporais são conhecidos previa-

mente. É atribuída a mesma prioridade para todas as ativações de uma mesma tarefa. O algoritmo estático mais conhecido e implementado é o escalonamento por taxas monotônicas (*Rate Monotonic Scheduling - RMS*). Este algoritmo atribui prioridades às tarefas de acordo com o número de vezes que serão executadas. Quanto maior for a frequência de execução, maior a prioridade da tarefa. Este método de escalonamento é preemptivo e é dito ótimo na classe dos escalonamentos de prioridade fixa (LIU, 2000).

- os algoritmos de escalonamento dinâmicos, ao contrário dos estáticos, não atribuem prioridades fixas às tarefas. A decisão de escalonamento é feita em tempo de execução e a prioridade das tarefas pode mudar. Os critérios para essa decisão variam dependendo do algoritmo. Um dos algoritmos de escalonamento dinâmico mais utilizado é o algoritmo *Earliest Deadline First (EDF)*. O EDF escolhe na fila de prontos a tarefa que tenha o *deadline* mais próximo do tempo atual. Se chegar na fila uma tarefa que tenha um *deadline* menor ainda, ocorrerá preempção. O EDF é um algoritmo ótimo na classe dos escalonamentos de prioridade dinâmica (LIU, 2000).

2.1.2 Criticalidade

Segundo Kopetz (2011), sistemas de tempo real podem ser classificados de acordo com o impacto gerado pela perda de um *deadline*. Existem três tipos:

- Críticos *(Hard)*: a perda de um *deadline* pode provocar falhas completas do sistema ou até mesmo catástrofes. Essas falhas podem representar danos irreversíveis em equipamentos ou ainda, em perda de vidas humanas. O sistema de controle de freios e o sistema de air-bag são exemplos de sistemas de tempo real críticos;

- Não-críticos *(Soft)*: a perda de um *deadline* implica em diminuição do desempenho mas não resulta em falhas ou danos muito significativos. As metas temporais descrevem o comportamento temporal desejado para o sistema. Exemplos: um editor de texto, sistema de folha de pagamento, sistema de processamento de matrículas, etc;

- Firmes *(Firm)*: a perda de um *deadline* não provoca falha total no sistema, porém, a perda de uma quantidade muito grande pode implicar

em falhas completas do sistema. Se uma tarefa não puder ser executada dentro do prazo máximo de tempo estipulado, ela deixa de ter valor para o sistema e deve ser abortada. Um sistema de controle de navegação de robôs móveis é um exemplo de um sistema tempo real firme.

2.1.3 Áreas de Aplicação

Sistemas de tempo real estão cada vez mais presentes no nosso dia a dia e são encontrados nas mais diversas áreas:

- Telecomunicações: Centrais telefônicas, videoconferência, aplicações multimídia, etc;
- Aeroespacial: aviação, satélites, etc;
- Defesa: controle de mísseis, radar, sonar;
- Setor financeiro: transações na bolsa de valores, sistemas bancários online;
- Transportes: sinalização ferroviária;
- Veículos: automação em aeronaves, automóveis, sondas espaciais;
- Indústria: controle de processos, robôs, etc;
- Entretenimento: videojogos, vídeo sob demanda, áudio, etc;

A Figura 2 mostra um exemplo de sistema de tempo real. Trata-se do *Curiosity Rover*, um robô móvel do tamanho aproximado de um carro médio, que foi enviado em 2011, pela *National Aeronautics and Space Administration* (NASA), para uma missão de exploração do planeta Marte. O robô é equipado com 12 câmeras no total, todas espalhadas nos mais variados ângulos e que enviam regularmente fotos de Marte. Além disso, o Curiosity também é equipado com um conjunto de equipamentos e sensores para análise de solo, umidade, temperatura, velocidade do vento e radiação, além de três antenas UHF responsáveis por transmitir e receber informações e um sistema modular de software otimizado para as missões. O robô possui seis rodas grandes e um forte sistema de suspensão para suportar a imprevisível e irregular superfície marciana (NASA, 2014).

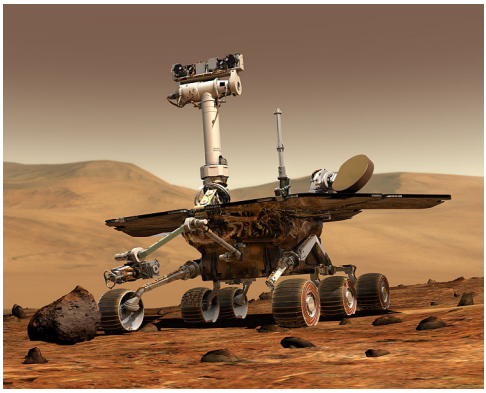

Figura 2: O robô de exploração a Marte Curiosity
Fonte: NASA

2.2 ROBÓTICA MÓVEL

A Robótica é o ramo da tecnologia e do conhecimento que lida com o projeto, construção, operação e aplicação de robôs em diversas atividades. Sistemas robóticos são constituídos por um conjunto de sistemas e equipamentos que auxiliam os robôs na execução de tarefas. A robótica é um campo multidisciplinar que utiliza as engenharias de Automação, Elétrica, Eletrônica, Mecânica além da Ciência da Computação, Biologia e outras disciplinas (DUDEK; JENKIN, 2010).

O termo robô vem originalmente da palavra *robota* do idioma checo, que significa 'trabalho forçado' (NEHMZOW, 2003). De acordo com Dudek e Jenkin (2010), um robô é um dispositivo ou conjunto de dispositivos eletromecânicos ou biomecânicos capazes de realizar tarefas de forma autônoma ou pré-programada. Um robô móvel é um robô capaz de se movimentar e interagir com um ambiente definido, através do uso de sensores e atuadores.

O uso tradicional da robótica consistia inicialmente em aplicações industriais que utilizam robôs fixos para controle de processos e manufatura, o que permitiu melhorar a qualidade dos produtos, aumentar a eficiência e produtividade, melhorar a segurança e reduzir os custos de produção. Posteriormente, e com a introdução da mobilidade na robótica, surgiu uma nova gama de aplicações, como por exemplo, a utilização de robôs móveis na realização de tarefas potencialmente perigosas para o ser humano, incluindo o transporte de cargas perigosas, manipulação de materiais tóxicos e explosivos, combate

a incêndios, patrulhamento e vigilância, etc. Neste sentido, a robótica móvel permitiu melhorar as condições de trabalho do homem, substituindo tarefas penosas por outras mais simples e vantajosas, além de melhorar a sua qualidade de vida.

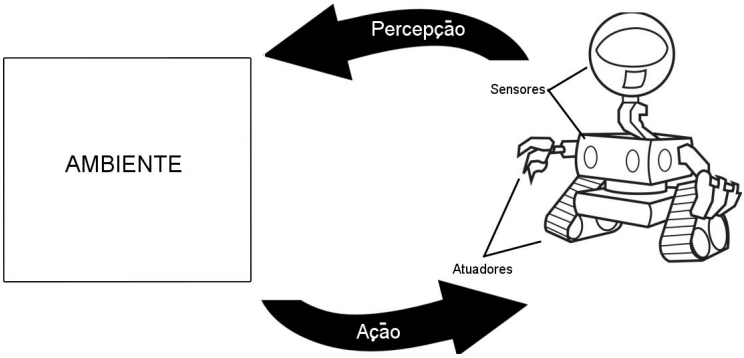

Figura 3: Ciclo de interação de um robô móvel com um ambiente

Por se tratar de um sistema de tempo real, o robô obtém informações sobre o ambiente através dos seus sensores e controla sua interação com este ambiente utilizando atuadores, que são responsáveis por executar ações. A Figura 3 ilustra o ciclo de interação de um robô móvel com um determinado ambiente através da percepção, realizada pelos sensores, e da ação, realizada pelos atuadores.

2.2.1 Classificação dos robôs

Os robôs podem ser classificados de acordo com a sua anatomia e forma de controle. Quanto à anatomia, podem ser aéreos, aquáticos e terrestres (PIERI, 2002). Os robôs móveis aéreos geralmente visam inspecionar grandes áreas e contam com câmeras de vídeo de boa definição. Os robôs aquáticos conseguem permanecer submersos na água a alguns metros de profundidade enquanto executam as suas tarefas. Já os terrestres, são os mais comuns e populares e realizam as suas tarefas em terra firme. Esses robôs podem ser classificados ainda em 3 grupos, dependendo do tipo de atuador utilizado:

- Robôs com rodas: mais simples, porém o desempenho em superfícies irregulares pode não ser bom;

- Robôs com esteiras: geralmente utilizados em terrenos irregulares, como por exemplo em solo mole ou cheio de pedras;
- Robôs com pernas: geralmente utilizados em ambientes acidentados com subidas ou ainda em ambientes com escadas;

Quanto à forma de controle, Siegwart, Nourbakhsh e Scaramuzza (2011) classificam os robôs móveis como tele-operados, semi-autônomos e autônomos. Um robô tele-operado se movimenta apenas obedecendo aos comandos de um operador. Já o semi-autônomo recebe do operador um macro comando a ser executado e o realiza sozinho. Ainda segundo Siegwart, Nourbakhsh e Scaramuzza (2011), o robô autônomo se movimenta e realiza tarefas de forma totalmente autônoma, sem assistência de um operador. Para tal, ele utiliza os dados obtidos do ambiente para tomar as próprias decisões. O grau de autonomia depende da capacidade do robô em abstrair o ambiente e converter a informação obtida em ações. O robô autônomo possui duas características que o diferenciam de outros veículos:

- Percepção: o robô utiliza os seus sensores para perceber o ambiente e gerar mapas globais e locais do mesmo.
- Raciocínio: para se mover e realizar a tarefa proposta, o robô deve ser capaz de planejar trajetórias seguras e poder modificá-las na presença de obstáculos inesperados.

2.2.2 Sensores

Conforme mencionado anteriormente, os robôs móveis autônomos possuem a capacidade de se deslocarem de forma autônoma em um ambiente conhecido ou desconhecido. O sistema de percepção de um robô móvel permite que este seja capaz de reagir mediante eventos imprevistos e mudanças no ambiente. Para tal, é necessário um sistema sensorial que consiga obter informações do ambiente em quantidade e qualidade suficientes para que o robô possa realizar a tarefa de maneira mais similar possível a um operador humano. Por esta razão, geralmente um robô é equipado com vários sensores que se complementam e facilitam a percepção do ambiente.

Aos elementos que desenvolvem a função de transformação de uma magnitude física em outra, chamamos de transdutores. Um sensor pode ser definido como sendo um transdutor que altera a sua característica física interna devido a um fenômeno físico externo (NEHMZOW, 2003).

De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), os sensores de um robô móvel podem ser classificados como:

- Sensores internos: são usados para medir posição, velocidade ou aceleração das juntas de um robô manipulador, ou das rodas de um robô móvel. Esta categoria de sensores inclui os sensores de posição, sensores de velocidade e sensores de aceleração;

- Sensores externos: são usados para monitorar o próprio robô e a sua relação com o ambiente durante a realização de uma tarefa. Fazem parte desta categoria os sensores de proximidade e sensores de distância.

É possível utilizar um robô sem qualquer sensoriamento externo, porém grande parte das aplicações necessita de sensores, como por exemplo a capacidade do robô móvel detectar obstáculos inesperados e desviar dos mesmos.

Entre os sensores utilizados pelos robôs móveis, Saha (2008) destaca os mais comuns:

- Sonares: o funcionamento de um sonar consiste na emissão e detecção de pulsos em uma frequência de som apropriada. Analisando o tempo necessário para que o pulso refletido seja detectado, é possível estimar a distância do objeto que produziu a reflexão relativamente ao sonar. São sensores baratos, porém muitas vezes podem ser imprecisos;

- Sensores de Infravermelho: funcionam de maneira similar aos sonares mas operam em frequências muito mais altas. Os mais simples são LEDs emissores de luz acoplados a receptores. São baratos e pequenos, porém muito ruidosos;

- Sensores Laser: o funcionamento de um laser consiste na emissão e detecção de pulsos de radiação laser de baixa potência. A acurácia deste tipo de sensor é consideravelmente superior aos sonares e infravermelhos. É possível se construir uma imagem com certa precisão que identifica as distâncias de cada ponto do obstáculo relativamente ao robô. Os sensores laser são muito caros para a maioria das aplicações (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011);

- Odometria: realizada por dispositivos acoplados aos motores ou eixo das rodas que conseguem estimar a distância percorrida pelo robô através da contagem do número de giros das rodas. É uma técnica barata e que proporciona uma boa precisão em curto prazo. Porém, ao longo do tempo é inevitável a acumulação de erros devido a imprecisões de manufatura de rodas, deslizamento em superfícies lisas e irregularidades do terreno;

- Sensores de choque: normalmente são formados por dois fios condutores, separados por uma pequena distância e colocados no perímetro do

robô. Conseguem detectar a colisão do robô com obstáculos quando o fio mais externo é empurrado na direção do fio interno, fechando um circuito e produzindo um sinal elétrico;

- Câmeras: a capacidade de visão é muito útil em robôs móveis. A maioria das câmeras em uso são baseadas na tecnologia *CCD (charged coupled device)*, em que a luz incide sobre uma matriz de capacitores fotossensíveis produzindo uma imagem. Muitas vezes são utilizadas duas ou mais câmeras montadas sobre um robô móvel, de modo a permitir visão com informação de profundidade;

- GPS: um dispositivo GPS *(Global Positioning System)* funciona recebendo sinais de satélites e combinando-os para estimar a posição do robô relativamente a um sistema de coordenadas.

2.2.3 Atuadores

Segundo Saha (2008), os dispositivos responsáveis pelo movimento e articulação das partes móveis de um robô móvel são chamados de atuadores. Eles executam as tarefas ordenadas pelo sistema de controle do robô. Dentre os vários atuadores presentes num robô móvel terrestre, se destacam os responsáveis pela sua locomoção:

- Rodas: são os atuadores mais usados em robôs móveis. Em geral, a locomoção baseada em rodas não apresenta bom desempenho em terrenos nos quais as irregularidades têm altura não desprezível quando comparadas ao raio das rodas;

- Hastes: são dispositivos que se assemelham a pernas ou patas de insetos. A locomoção através de hastes é interessante em robôs quando estes se deslocam em ambientes específicos, como aqueles com escadas, regiões muito íngremes, etc. O custo desses sistemas pode ser alto, uma vez que pelo menos dois motores são necessários em cada haste;

- Lagartas: são dispositivos que consistem numa esteira que se acopla às rodas de um robô terrestre com a finalidade de aumentar a aderência ao solo. A locomoção baseada em lagartas é interessante nos casos em que o robô deve se mover em terrenos acidentados ou pouco estáveis. Por outro lado, este tipo de locomoção não costuma ser muito eficiente devido à fricção das rodas dentro das lagartas e a dissipação de energia resultante do deslizamento das esteiras durante os movimentos de giro.

2.2.4 Arquitetura de robôs móveis

A escolha da arquitetura de um robô móvel reflete diretamente na forma como este robô vai se comportar e realizar a tomada de decisões. As arquiteturas podem ser classificadas em reativas e deliberativas. Um terceiro tipo surge com a combinação das duas anteriores, os sistemas híbridos. Esta nova arquitetura tenta explorar as vantagens dos dois mecanismos e amenizar as suas desvantagens (HOLLAND, 2004).

A arquitetura reativa é baseada no comportamento animal. O robô reage diretamente aos estímulos externos. É composta por um conjunto de comportamentos que relacionam certas condições sensoriais a um conjunto de ações do robô. Apresenta boa performance em termos de requisitos temporais, porém a incerteza do sucesso da tarefa e a dificuldade de definição do conjunto mínimo de comportamentos para tarefas mais complexas constituem desvantagens significativas. Em geral, esta arquitetura produz robôs adequados para operações em tempo real (BROOKS, 1985).

Ainda de acordo com Brooks (1985), a arquitetura deliberativa incorpora um estágio intermediário de planejamento entre o sensoreamento e a ação. Antes da seleção de qualquer ação, há a execução de um plano. As informações obtidas pelos sensores são guardadas numa estrutura de dados global, que é acessada pelo estágio de planejamento. O robô constrói um modelo interno que representa o ambiente em que está inserido e antes de realizar uma tarefa ele tenta explorar o espaço de soluções obtidas a partir do modelo interno. Assim, o robô consegue gerar um plano de atividades e só então executa alguma ação através dos seus atuadores. Uma limitação desta arquitetura é que para o planejamento ser ótimo, é preciso ter um bom modelo do ambiente e a construção deste modelo é custoso.

Por fim, a arquitetura híbrida incorpora um elemento de planejamento sobre os comportamentos reativos. Esta arquitetura corresponde a uma arquitetura reativa controlada por um plano de execução e sequenciamento de comportamentos. É possível fazer a reconfiguração dinâmica de sistemas de controle reativo através da incorporação da habilidade de raciocínio baseado em modelos do ambiente. A integração de deliberação e controle reativo é um problema complexo, mas permite produzir sistemas mais flexíveis, robustos e inteligentes (ROMERO, 2011). Para que um robô móvel com esta arquitetura possa se locomover num ambiente conhecido e realize as suas tarefas de forma segura, geralmente ele utiliza um mapa com alguma representação deste ambiente. Este mapa contém informações importantes sobre o ambiente tais como áreas inacessíveis, obstáculos conhecidos e outras informações relevantes. Para conseguir alcançar uma posição do ambiente a partir de outra, o robô faz o planejamento da rota de forma a obter o melhor trajeto possível,

evitando obstáculos conhecidos. Durante a trajetória, caso sejam detectados obstáculos desconhecidos através dos sensores, o robô faz um novo planejamento de rota de forma a poder se desviar de tais obstáculos (HOLLAND, 2004).

As formas de representação de um ambiente conhecido, bem como técnicas de navegação e planejamento de rota são abordados no próximo capítulo deste trabalho.

2.3 CONSIDERAÇÕES FINAIS

A ideia principal deste trabalho consiste em aplicar o conceito de restrição temporal, inerente aos Sistemas de Tempo Real, na realização de tarefas de navegação dos robôs móveis. Este capítulo apresentou conceitos básicos dos Sistemas de Tempo Real, com especial foco na existência de um *deadline* firme que deve ser cumprido pelas tarefas e nas consequências que a perda deste *deadline* pode trazer para o sistema. Também foram apresentados conceitos básicos de robótica móvel, fundamentais para a compreensão do restante do trabalho.

As tarefas em que o mecanismo proposto será aplicado consistem em mover o robô de uma posição inicial até uma posição destino e na volta para a posição inicial, dentro de um ambiente parcialmente conhecido. Neste sentido, o próximo capítulo faz um levantamento sobre os principais fundamentos da localização de um robô móvel dentro de um determinado ambiente e as formas como o robô percebe este ambiente e navega pelo mesmo.

3 NAVEGAÇÃO DE ROBÔS MÓVEIS

Este capítulo descreve as principais técnicas de localização de um robô móvel, modelagem do seu ambiente, técnicas de navegação e planejamento de trajetória, presentes na literatura. O capítulo não visa abordar os últimos algoritmos disponíveis na literatura, mas sim as técnicas e fundamentos nos quais estes algoritmos se baseiam.

De acordo com Leonard e Durrant-Whyte (1991), o problema da navegação de um robô móvel, na realização de tarefas, pode ser resumido em três questões importantes:

- Onde estou?
- Para onde vou?
- Como devo chegar lá?

O problema da localização consiste em responder à primeira questão, do ponto de vista do robô. Antes de um robô conseguir planejar rotas e se locomover, é necessário um sistema de localização de forma que seja possível identificar em que ponto do ambiente o robô se encontra num determinado instante de tempo. Conhecendo o ambiente, a localização do robô e a posição de destino, o planejamento de trajetória permite determinar uma rota segura da posição de origem até o destino, evitando possíveis obstáculos no caminho.

3.1 LOCALIZAÇÃO DE ROBÔS MÓVEIS

A localização de um robô móvel consiste em conhecer sua posição e sua orientação em relação a algum referencial fixo no ambiente. Para Thrun et al. (2000), o problema da localização é a chave para o sucesso de um sistema robótico autônomo. Borenstein, Everett e Feng (1996) classificam os métodos de localização de um robô como métodos de localização relativa ou métodos de localização absoluta.

3.1.1 Localização relativa

Métodos desta técnica de localização utilizam unicamente os sensores e atuadores do próprio robô para estimar sua localização. Esta estimativa é baseada nas localizações estimadas nos instantes anteriores, o que geralmente

implica em propagação e acúmulo de erros ao longo do tempo. A estimativa da localização pode ser feita de duas formas:

- Odometria: uma das técnicas de localização mais utilizadas devido ao seu baixo custo e boa precisão em pequenos trechos percorridos (BORENSTEIN; EVERETT; FENG, 1996). A estimativa da posição é feita através da leitura de dispositivos de locomoção do robô a partir de um ponto inicial. Um dos exemplos mais comuns do uso desta técnica, é um sistema odométrico que monitora as rotações das rodas de um robô. Neste caso, a localização do robô é determinada pela integração incremental do movimento das rodas. É possível calcular a distância percorrida e a orientação do robô através do uso de sensores de rotação, que contam a quantidade de giros em cada roda. A desvantagem desta técnica é o acúmulo de erros que aumenta proporcionalmente à distância percorrida, fazendo com que o erro causado em um determinado instante se propague para os instantes seguintes. Após várias medidas, a posição real do robô pode ser muito diferente da estimada. Para melhorar a precisão da localização, a odometria é muitas vezes utilizada em conjunto com métodos de localização absoluta. A técnica da odometria é atrativa porque, além de barata, é autocontida, ou seja, não necessita de informações externas para estimar a localização do robô (BORENSTEIN; EVERETT; FENG, 1996).

- Navegação inercial: esta técnica utiliza as leituras de velocidade e aceleração de um robô para estimar a sua localização atual. Assim como na odometria, esta técnica consegue estimar a localização do robô a partir da integração das informações dos sensores (neste caso, giroscópios e acelerômetros) (THRUN et al., 2000). É um método atrativo por ser auto-contido, porém erros nos sensores podem causar erros com crescimento ilimitado nas estimativas. Da mesma forma que sistemas odométricos, correções de estimativas podem ser realizadas por sistemas auxiliares. É um método muito utilizado em dispositivos que realizam grandes deslocamentos em curtos períodos de tempo, como por exemplo aviões, mísseis, foguetes, etc.

3.1.2 Localização absoluta

Os métodos de localização absoluta necessitam de pontos de referência externos ao robô e conseguem informar a localização do robô, independentemente de localizações calculadas anteriormente. A partir dos pontos de referência e de suas posições conhecidas, o robô é capaz de estimar sua posição.

Ao contrário dos métodos de localização relativa, os erros de medição na localização absoluta não são cumulativos (PEDROSA, 2006). A determinação da localização do robô pode ser feita de diferentes modos:

- Sinais ativos: nesta técnica, bases estáticas são utilizadas para emitir sinais direcionados até um ponto qualquer do ambiente. Através da leitura desses sinais, é possível estimar a localização do robô. Esses sinais ativos podem ser sinais de satélite, ondas de rádio, lasers, luzes, etc. A localização por sinais ativos é muito utilizada na navegação aérea e marinha. Vários tipos de equipamentos podem ser usados, como os baseados em sensores de ultra-som e o *GPS*;

- Marcas artificiais: nesta técnica, objetos (ou marcas) são colocados estrategicamente em locais conhecidos do ambiente. Ao encontrar esses objetos, o robô consegue se orientar e estimar a sua posição. É comum o uso de códigos de barra e figuras geométricas coloridas facilmente identificáveis no ambiente;

- Marcas naturais: esta técnica possui funcionamento similar à técnica de marcas artificiais. A única diferença é que os objetos são nativos do ambiente. Em ambientes internos, as marcas podem ser portas, janelas, quinas, luz no teto, etc e em ambientes externos podem ser pedras, árvores, etc. O ambiente precisa ser previamente conhecido (PEDROSA, 2006).

- Mapas: Nas situações em que o mapa do ambiente é conhecido pelo robô, o mesmo compara as informações obtidas através dos sensores com as informações do mapa. A partir da comparação, a posição absoluta do robô pode ser deduzida caso as leituras dos sensores coincidam com o modelo do mapa. Uma desvantagem desta técnica é a necessidade de uma grande quantidade de informações sensoriais para realizar a comparação e o alto custo computacional (NEGENBORN, 2003). De acordo com Thrun et al. (2000), a correta localização de um robô relativamente a um sistema de coordenadas global é um dos pré-requisitos para a construção de um mapa. A qualidade do mapa depende fortemente do sistema de localização. Os erros acumulados na estimativa da localização do robô podem causar uma interpretação incorreta dos dados sensoriais e consequentemente a construção de um mapa incorreto.

3.2 MODELAGEM DO AMBIENTE

De acordo com Mataric (1992), para que um robô móvel consiga interagir de forma eficaz com o ambiente em que está inserido e não se limite apenas à navegação aleatória, é necessário criar alguma forma de representação (ou modelo) deste ambiente. Esta representação é construída a partir das leituras obtidas pelos sensores e é utilizada no planejamento de trajetória, de modo que o robô consiga coordenar suas ações e realizar suas tarefas. O modelo do ambiente permite que o robô possa lidar, de modo mais flexível, com situações inesperadas que eventualmente possam acontecer. Ao processo de construção de um modelo do ambiente físico de um robô, chamamos de mapeamento e o resultado deste processo é um mapa do ambiente.

Na robótica móvel, existem duas abordagens principais para a representação de ambientes: a abordagem topológica e a abordagem métrica (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Na primeira abordagem, o ambiente é visto como um grafo, onde os vértices geralmente representam os espaços livres do ambiente e as arestas indicam informações de conectividade entre os espaços, como por exemplo, a distância entre os mesmos. Nesta abordagem, o planejamento de trajetórias é eficiente e com baixo custo computacional. Além disto, não é necessária a localização precisa do robô (FABRIZI; SAFFIOTTI, 2000).

A abordagem métrica produz uma definição geométrica do ambiente no qual o robô está inserido. O ambiente é apresentado com alta riqueza de detalhes, incluindo a posição dos objetos que o compõem, como portas, paredes, obstáculos, armários, etc. Por ser baseada na geometria, esta abordagem favorece o reconhecimento de lugares e facilita a computação de caminhos curtos. Por outro lado, requer a determinação precisa da localização do robô e é menos eficiente no planejamento de trajetórias pois apresenta custo computacional relativamente alto (PEDROSA, 2006).

3.3 TÉCNICAS DE NAVEGAÇÃO

Na robótica móvel, a navegação corresponde à habilidade de um robô se locomover em um determinado ambiente, de uma posição inicial até uma final, fazendo uso das informações parciais de sua posição e do ambiente em que está inserido. Trata-se de um dos principais problemas enfrentados no desenvolvimento de robôs móveis e envolve tarefas básicas como modelagem do ambiente e planejamento de trajetória (GE, 2006).

Como visto anteriormente, a modelagem do ambiente consiste na representação (ou modelo) que o robô possui do ambiente em que está inserido.

Esta representação é obtida através do sistema sensorial ou já é conhecida previamente. As técnicas de navegação podem ser divididas em técnicas de abordagem reativa (ou sensorial) ou técnicas de abordagem representativa.

3.3.1 Abordagem reativa

Na abordagem reativa, a navegação é baseada exclusivamente nos sensores do robô. Nesta abordagem, o ambiente não é conhecido previamente e pode estar em constante mudança. A cada instante, o robô realiza leituras dos seus sensores para modificar a sua rota.

Várias técnicas de navegação são consideradas reativas, entre elas, as baseadas nos algoritmos da família *Bug*. De acordo com Kshirsagar e Shukla (2011), esses algoritmos supõem que o robô é um ponto no espaço e seus sensores são precisos. Estes algoritmos não realizam nenhuma forma de representação do ambiente. A cada instante em que os dados dos sensores são processados pelos controladores, o robô atualiza sua rota. Trata-se de algoritmos de fácil implementação e com baixo custo de memória. O algoritmo da família *Bug* mais simples é o algoritmo *Bug 1*. A técnica deste algoritmo consiste em navegar na direção do objetivo a ser alcançado. Caso um obstáculo seja encontrado no caminho, a sua borda deve ser contornada pelo robô até que o objetivo esteja novamente acessível. O algoritmo tenta encontrar o ponto da borda do obstáculo mais próximo ao destino final e só então se afasta do obstáculo e segue em direção ao objetivo (HOLLAND, 2004).

Outro algoritmo da família *Bug* é o algoritmo *Bug 2*. O funcionamento deste algoritmo é similar ao do algoritmo anterior. A diferença é que uma reta imaginária é traçada do ponto de origem até o destino e o robô não precisa contornar o obstáculo por inteiro, apenas até encontrar um ponto da reta traçada. Além dos dois algoritmos citados anteriormente, Holland (2004) cita vários outros da família *Bug*: *DistBug, TangentBug, WedgeBug e RoverBug*. Cada um deles tenta resolver os casos onde os outros falham, mas acabam introduzindo novos problemas. Estes algoritmos geralmente não são utilizados como estratégia principal de navegação em robôs móveis modernos (GE, 2006).

3.3.2 Abordagem representativa

Na abordagem representativa, o modelo do ambiente é conhecido previamente e por esta razão, é possível empregar técnicas de planejamento de

trajetórias para encontrar a melhor rota possível a ser seguida.

O processo de planejamento de trajetória atribui aos robôs a capacidade de idealizar e planejar seus próprios movimentos, sem a necessidade da interferência humana. Neste contexto, considera-se o robô como sendo um único objeto rígido, não apresentando partes móveis, como braços e pernas. Considera-se também que a movimentação do robô é limitada apenas pelos obstáculos dispostos no ambiente.

Segundo Jahanbin e Fallside (1988), o problema do planejamento de trajetória pode ser definido como a busca de um percurso a ser seguido pelo robô e a sequência de ações que o mesmo deve executar para que ele possa alcançar uma posição destino, evitando a colisão com o conjunto de obstáculos do ambiente.

O método de planejamento de trajetória apresenta duas vertentes principais: o planejamento global e o planejamento local. Ambos podem ser utilizados simultaneamente para melhorar o desempenho do robô em um ambiente real.

No planejamento de trajetória global, o modelo do ambiente onde o robô está inserido é um modelo simplificado, estático e pré-gravado. Este modelo permite que seja traçada uma trajetória que pode passar por locais que o robô não consegue perceber com seus sensores, seja por estarem fora do alcance ou por estarem obstruídos por algum obstáculo. Neste tipo de planejamento de trajetória, os obstáculos do ambiente são conhecidos e considerados fixos. Já o planejamento local é responsável pela navegação curta, baseada nos valores coletados pelos sensores do robô. Durante a navegação do robô, caso o ambiente seja dinâmico, obstáculos desconhecidos fixos e/ou móveis podem ser encontrados. O planejamento local faz uso das informações obtidas pelos sensores para evitar tais obstáculos e manter a integridade do robô. Os robôs modernos fazem uso das duas vertentes do planejamento de trajetória descritas anteriormente, para obter melhores resultados na navegação (GE, 2006).

3.4 TÉCNICAS DE PLANEJAMENTO DE TRAJETÓRIA

Entre as diferentes técnicas para se fazer o planejamento de trajetória, as principais são os *Roadmaps*, as Decomposições em Células e a técnica dos Campos Potenciais.

3.4.1 Roadmaps

Esta técnica reduz as informações de um ambiente a um grafo que representa os possíveis caminhos. Cada vértice do grafo corresponde a um local específico do ambiente e uma aresta corresponde a um caminho entre locais vizinhos. O planejamento de trajetórias se resume a conectar as posições iniciais e finais do robô ao grafo e buscar neste um caminho entre os dois pontos. Geralmente, métodos roadmaps são rápidos e simples de se implementar, porém não fornecem uma boa representação das informações do ambiente. Existem vários métodos baseados nesta abordagem. Entre os principais estão os grafos de visibilidade e os diagramas de Voronoi (CHOSET, 2005).

- **Grafos de Visibilidade:** um dos mais antigos métodos Roadmap é o método de Grafos de Visibilidade, que foi proposto inicialmente para ambientes bidimensionais com objetos poligonais. De forma simplificada, um grafo de visibilidade consiste em um conjunto de todos os possíveis caminhos de um ponto de origem até o destino que o robô pode seguir em um determinado ambiente, sem colidir com obstáculos (LATOMBE, 1991).

  O grafo de visibilidade se baseia no conjunto dos vértices dos polígonos e de sua visibilidade mútua. Deste modo, este grafo contém todos os menores trajetos existentes no ambiente.

  O grafo é obtido gerando-se segmentos de reta entre os pares de vértices dos obstáculos. Todos os segmentos de reta que estiverem inteiramente na região do espaço livre são adicionados ao grafo. No planejamento de trajetória, as posições de origem e destino são representadas como vértices, de modo a gerar um grafo de conectividade onde um algoritmo de busca pode ser utilizado para encontrar uma rota livre.

  Para que esta técnica possa ser utilizada, é necessário que o mapa seja completo e bem definido. O ambiente não pode ter objetos móveis e a localização do robô móvel deve ser conhecida com precisão durante toda a navegação (LATOMBE, 1991).

- **Diagramas de Voronoi:** um diagrama de Voronoi pode ser definido como uma estrutura geométrica que representa informações de proximidade sobre um conjunto de pontos ou objetos. Dada uma série de sítios (também chamados geradores) ou objetos, o plano é particionado atribuindo para cada ponto o seu sítio mais próximo.

  Todos os pontos que não possuírem um único sítio mais próximo formam o diagrama de Voronoi. Num diagrama de Voronoi, os pontos

são equidistantes de dois ou mais sítios. O grafo roadmap desta técnica consiste em arestas Voronoi que constituem a trajetória.

No processo de planejamento de trajetória, como a região onde o robô se locomove geralmente contém obstáculos, cada um destes obstáculos pode ser representado por polígonos côncavos ou convexos. Para o cálculo do diagrama para essa coleção de polígonos, é feita uma aproximação, convertendo os obstáculos em uma série de pontos. Em seguida, se calcula o diagrama de Voronoi para esta coleção de pontos. Por fim, os segmentos do diagrama que intersectam algum obstáculo são eliminados (BRAUNL, 2008).

3.4.2 Decomposição em Células

O método de decomposição em células consiste em dividir o espaço livre do robô em regiões simples (ou células), de forma que um caminho entre duas configurações em células diferentes possa ser facilmente gerado.

Um grafo não direcionado que representa a relação de adjacência entre as células é construído, sobre o qual é realizada a busca da trajetória. Este grafo é chamado grafo de conectividade e os seus vértices são as células extraídas do espaço livre do robô. Dois vértices estão conectados por uma aresta se e somente se as células correspondentes a eles são adjacentes. O resultado da busca efetuada é uma sequência de células denominada canal. Um caminho contínuo pode ser computado a partir do canal (BRAUNL, 2008).

Os métodos baseados em decomposição em células podem ser divididos em exatos e aproximados. Os métodos exatos decompõem o espaço livre em um conjunto de células cuja união é exatamente o espaço livre do ambiente. Caso exista um caminho entre duas configurações quaisquer, ele sempre pode ser obtido, desde que seja utilizado um algoritmo de busca apropriado.

Métodos aproximados de decomposição em células dividem o espaço livre em um conjunto de células de forma predefinida cuja união está estritamente contida no espaço livre. Esses métodos podem não ser completos, pois dependendo da precisão utilizada, não é possível encontrar um caminho entre duas configurações, mesmo que este exista. Contudo, por serem mais simples, são utilizados com maior frequência na prática.

3.4.3 Campos potenciais

Este método consiste em discretizar o espaço de configuração em uma fina grade regular de configurações e realizar a busca por um caminho livre

nesta mesma grade. A cada posição desta grade é associado um valor de uma função, com a qual pode-se fazer a analogia de um campo potencial. A grade resultante pode ter um tamanho muito grande. Neste caso, são utilizados métodos heurísticos para encontrar um caminho no amplo espaço de buscas.

Pode-se dizer que neste método, os obstáculos exercem uma força repulsiva sobre o robô enquanto que o objetivo aplica uma força atrativa sobre o mesmo. A força resultante, a soma de todas as forças, permite calcular o sentido da trajetória e a velocidade do movimento. Os métodos de campos potenciais podem ser facilmente implementados e produzem um resultado inicial aceitável sem a necessidade de muitos refinamentos. Por outro lado, apresentam uma série de problemas inerentes, que independem de uma implementação em particular: situações de armadilha devido a mínimos locais (comportamento cíclico) (CHOSET, 2005).

3.5 CONSIDERAÇÕES FINAIS

O mecanismo de previsão proposto neste trabalho funciona juntamente com os sistemas de localização, planejamento de trajetória e navegação do robô. Neste sentido, este capítulo apresentou uma revisão da literatura sobre as principais técnicas de localização de um robô móvel, as formas de representação de um ambiente e as técnicas de planejamento de trajetória e de navegação neste mesmo ambiente.

Na posse de um modelo do ambiente, grafos/grades são construídos a partir das informações deste ambiente (espaços livres, caminhos livres e obstáculos). O planejamento de trajetória se resume a algoritmos de busca nos grafos/grades, com o objetivo de se obter um trajeto seguro para o robô, entre duas posições no ambiente.

O próximo capítulo aborda a proposta desta dissertação, apresentando a arquitetura do sistema e descrevendo o mecanismo de previsão em maiores detalhes.

4 PROPOSTA

A proposta desta dissertação é a definição de um mecanismo de previsão de perda de *deadline* para robôs móveis autônomos, na realização de tarefas de navegação. Neste capítulo são apresentados a arquitetura do sistema proposto e o seu funcionamento.

4.1 DESCRIÇÃO INICIAL

Dada uma determinada tarefa de navegação e um tempo máximo de execução, o mecanismo deve ser capaz de prever em tempo hábil se o robô será capaz de concluir a tarefa dentro do prazo estipulado, com uma taxa de acerto desejável de 90% ou superior. Os mecanismos de controle presentes na camada inferior de baixo nível dos robôs móveis e de outros sistemas de tempo real possuem recursos que permitem uma tolerância de perda de *deadlines* de até 10% (FARINES; FRAGA; OLIVEIRA, 2000). Tais recursos da camada inferior permitem ao mecanismo de previsão de perda de deadline, que é executado numa camada superior de alto nível, tolerar perdas de *deadlines* também na faixa dos 10%.

O resultado da previsão deve ser fornecido na metade da trajetória do robô, de modo que em caso de previsão positiva de perda de *deadline*, a tarefa possa ser cancelada ou outras ações possam sem executadas. Como o ambiente não é totalmente conhecido e o mesmo pode ser dinâmico, o resultado da previsão de perda de *deadline* não deve ser calculado no início da tarefa pois neste instante o mecanismo desconhece a existência de possíveis obstáculos desconhecidos ao longo da trajetória do robô, durante a realização da tarefa. Por outro lado, a previsão não deve ser realizada no final da tarefa pois o resultado não seria fornecido a tempo da aplicação executar ações corretivas para minimizar as consequências da perda do *deadline*. Assim sendo, a metade da tarefa se torna um momento adequado para o cálculo da previsão.

Para a elaboração do mecanismo e consequentemente a sua avaliação, é necessário que se estabeleça um modelo de tarefas em que o mecanismo é aplicado. A tarefa considerada neste trabalho consiste na locomoção do robô de uma posição de origem até uma posição objetivo dentro de um ambiente parcialmente ou totalmente conhecido, e na volta à posição inicial conforme mostrado na Figura 4. Um ambiente totalmente conhecido é estático, isto é, nunca sofre modificações e o robô possui uma representação completa (um mapa) deste ambiente abrangendo todas as suas regiões e obstáculos presentes. Já num ambiente parcialmente conhecido, o robô também pos-

sui um mapa descrevendo suas informações, porém o ambiente pode sofrer modificações desconhecidas pelo robô. Neste tipo de ambiente, durante a realização das tarefas, obstáculos desconhecidos podem ser encontrados, os quais terão influência direta no resultado da previsão.

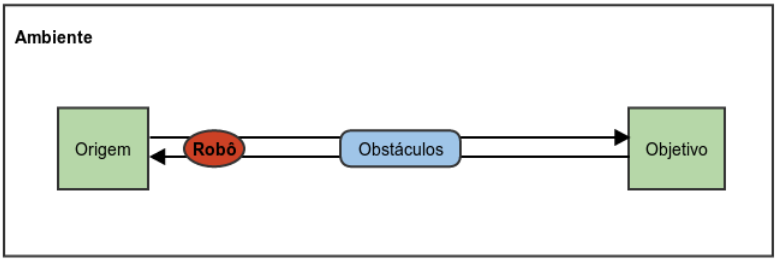

Figura 4: Modelo de tarefa em que o mecanismo pode ser aplicado

O funcionamento do mecanismo proposto consiste na coleta de informações durante a realização de tarefas e comparação com dados armazenados de tarefas anteriores. Um histórico de dados coletados é armazenado e consultado no processo de previsão.

O mecanismo funciona em dois modos distintos: o modo de aprendizagem e o modo de previsão. O modo de aprendizagem é utilizado quando o tamanho do histórico não é grande o suficiente para produzir resultados confiáveis, ou seja, para atingir uma taxa de previsões corretas igual ou superior a 90%. Neste caso, tarefas devem ser realizadas pelo robô enquanto o histórico é alimentado até atingir o tamanho mínimo estipulado. O modo de previsão é o modo principal e é utilizado quando já existem informações suficientes armazenadas no histórico.

O algoritmo de previsão de perda de *deadline* se traduz em uma fórmula cujo resultado indicará se determinada tarefa será ou não concluída dentro de um prazo estipulado. A partir disso, métricas irão avaliar a qualidade das previsões.

4.2 ARQUITETURA DO SISTEMA

O mecanismo de previsão de perda de *deadline* proposto nesta dissertação pode ser aplicado em qualquer robô móvel terrestre que não seja um sistema crítico e utilize uma arquitetura híbrida. Um robô com esta arquitetura utiliza um mapa com alguma representação do ambiente, que pode ser par-

cialmente ou totalmente conhecido. Este mapa contém informações importantes sobre o ambiente tais como áreas inacessíveis, obstáculos conhecidos e outras informações relevantes. Para conseguir alcançar uma posição do ambiente a partir de outra, inicialmente é feito um planejamento global de trajetória de forma a se obter o menor trajeto possível entre as duas posições, evitando obstáculos conhecidos presentes no mapa do ambiente. Durante a navegação do robô, caso sejam detectados obstáculos desconhecidos (ausentes no mapa do ambiente), é feito um novo planejamento local de trajetória de forma que o robô possa desviar de tais obstáculos. O planejamento local considera o menor desvio possível do obstáculo detectado e retorno à trajetória original. O mecanismo de previsão faz uso dos sistemas de planejamento de trajetória, navegação, localização e detecção de obstáculos, não dependendo de nenhuma implementação específica destes. No caso do sistema de planejamento de trajetória, basta que este seja capaz de retornar um conjunto de pontos (x,y) que representam a trajetória entre duas posições no ambiente, não sendo relevante a implementação interna do algoritmo utilizado para gerar a trajetória. No caso do sistema de navegação, basta que o mesmo seja capaz de enviar comandos ao robô para que a trajetória gerada pelo sistema de planejamento de trajetórias seja cumprida de forma mais fiel possível. Além disso, deve indicar ao mecanismo de previsão o momento de início e conclusão de uma tarefa. Para o sistema de localização, o mecanismo de previsão precisa de uma informação que representa a localização do robô e de obstáculos no ambiente, no formato (x,y). O método utilizado internamente para a localização não é relevante para o mecanismo. O sistema de detecção de obstáculos deve ser capaz de indicar a presença de obstáculos desconhecidos ao longo da trajetória do robô, a partir da leitura dos sensores. Os sensores utilizados para este fim podem variar. No entanto, quanto maior for a precisão do sensor utilizado, mais precisa será a detecção de obstáculos. O laser é uma excelente opção quando comparado a sonares e infravermelhos, porém seu custo é consideravelmente superior aos demais. Uma câmera de video também pode ser utilizada para a detecção de obstáculos.

A Figura 5 ilustra a arquitetura geral do sistema proposto.

O **módulo de localização** é o módulo central do sistema e é utilizado por todos os demais. Este módulo é responsável por localizar o robô dentro de um determinado ambiente, atribuindo uma posição ao robô. Uma posição dentro de um determinado ambiente depende de como este ambiente é representado. Tipicamente, numa representação métrica em duas dimensões, uma posição consiste em uma tupla (x,y,θ), sendo x e y as coordenadas cartesianas em relação a um eixo de referência e θ o ângulo de orientação. Todos os demais módulos dependem deste pois sem a correta localização do robô no ambiente não é possível fazer o planejamento de trajetória e a navegação

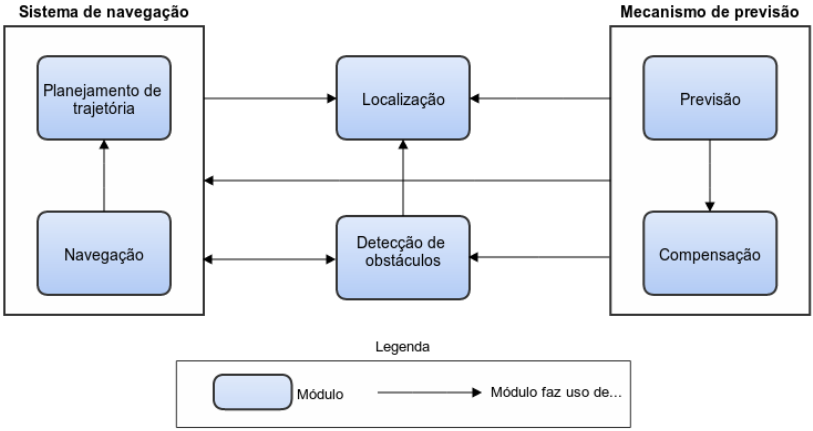

Figura 5: Arquitetura do sistema na visão de módulos

do robô de forma correta. Consequentemente, o mecanismo de previsão não produziria resultados confiáveis nestas condições.

O **sistema de navegação** é composto pelo módulo de planejamento de trajetória e o módulo de navegação. O **módulo de planejamento de trajetória** é responsável por gerar a trajetória mais curta possível entre duas posições no ambiente, de forma a evitar os objetos conhecidos e as áreas proibidas do ambiente. Para tal, o módulo utiliza uma representação do ambiente, como por exemplo, um mapa. Este módulo retorna um conjunto de posições por onde o robô deverá passar até chegar na posição destino.

O **módulo de navegação** é o módulo responsável pelo cumprimento da trajetória. Este módulo utiliza o módulo de planejamento de trajetória para obter a trajetória que o robô deve seguir e monitora a navegação, enviando constantemente comandos ao robô para que a trajetória do mesmo seja o mais fiel possível à trajetória planejada. Faz uso do módulo de detecção de obstáculos para detectar a presença de possíveis obstáculos inesperados ao longo da trajetória planejada. Caso ocorra uma detecção, o módulo de navegação obtém a trajetória de desvio do obstáculo através do módulo de planejamento de trajetória e instrui o robô de forma a se desviar dos obstáculos. Os comandos que o módulo envia ao robô incluem a velocidade certa para a posição ocupada pelo robô no ambiente, o ângulo de orientação, etc.

O **módulo de detecção de obstáculos** é o módulo responsável por monitorar as leituras dos sensores do robô com o objetivo de detectar a presença de obstáculos inesperados (fixos e móveis) ao longo da trajetória do robô. O

módulo obtém a trajetória planejada através do módulo de planejamento de trajetória. Quanto maior for a precisão dos sensores utilizados, maior a precisão na detecção de obstáculos. Geralmente, o laser se apresenta como a melhor opção por se tratar de um sensor com grande alcance e precisão muito alta. Sensores como sonares, infravermelhos e até câmeras podem ser utilizados para este fim, porém a detecção de obstáculos pode ficar comprometida pela imprecisão e ruídos na leitura dos mesmos (PIERI, 2002).

O módulo de previsão e o módulo de compensação constituem o mecanismo de previsão. O **módulo de previsão** é o módulo principal do mecanismo e responsável por fazer a previsões de perda de *deadline*. O **módulo de compensação** monitora as execuções de tarefas do robô com o objetivo de detectar mudanças bruscas e repentinas no ambiente que podem alterar o resultado das previsões. O objetivo do módulo é compensar essas mudanças inesperadas de modo que o resultado da previsão não seja incorreto. O mecanismo de previsão utiliza o sistema de navegação do robô para obter informações importantes sobre as tarefas, como por exemplo, a posição inicial e objetivo, o momento do início e de conclusão da tarefa, o momento da chegada do robô na posição objetivo, informações conhecidas do ambiente, etc. O mecanismo também faz uso do módulo de detecção de obstáculos para obter informações sobre o número de obstáculos desconhecidos encontrados pelo robô durante a realização da tarefa. A próxima seção deste capítulo aborda o mecanismo de previsão em maiores detalhes.

4.3 O MECANISMO DE PREVISÃO

Conforme mencionado anteriormente, o mecanismo proposto faz a previsão da perda de *deadline* de uma tarefa através da comparação dos dados atuais desta tarefa com dados armazenados de tarefas anteriores similares, ou seja, tarefas com a mesma posição inicial e objetivo, mesma quantidade de obstáculos conhecidos no ambiente, ocupando posições iguais ou próximas às posições já conhecidas. Durante a realização de qualquer tarefa, obstáculos desconhecidos e inesperados podem ser encontrados pelo robô.

O mecanismo utiliza um histórico principal para armazenar as informações das tarefas e o representa através de uma estrutura de dados que é consultada durante o processo de previsão. O mecanismo também faz uso de um histórico adicional, chamado de histórico de compensação e este será descrito mais à frente neste capítulo.

4.3.1 Histórico principal do mecanismo

Os dados de tarefas já executadas são armazenados no histórico principal do mecanismo e são utilizados na previsão de perda de *deadline* de tarefas posteriores.

O histórico principal tem um tamanho máximo definido e sempre que este tamanho é alcançado, as informações mais antigas são substituídas por informações mais recentes. O histórico armazena informações como a posição de origem do robô e a posição objetivo, o tempo parcial de execução da tarefa (tempo para atingir a posição objetivo), o tempo total de execução da tarefa (quando o robô retorna à posição inicial), a quantidade de obstáculos conhecidos no ambiente e as suas posições e a velocidade máxima do robô. Sempre que uma tarefa é concluída, o histórico é atualizado com as suas informações, caso nenhum obstáculo desconhecido tenha sido encontrado ao longo da trajetória do robô. Caso contrário, a informação é armazenada no histórico de compensação.

O histórico principal é consultado assim que o robô recebe a indicação de uma nova tarefa (o módulo de previsão obtém esta informação através do módulo de planejamento de trajetória), antes do robô começar a se locomover. No modo de aprendizagem, simuladores de robôs móveis podem ser utilizados para alimentar o histórico principal até o mesmo atingir o tamanho desejado.

Um exemplo do histórico principal do mecanismo é mostrado na Tabela 1, onde:

- **Origem** é a posição inicial do robô.
- **Objetivo** é a posição objetivo que deve ser atingida.
- **TP** é o tempo parcial de execução em segundos (tempo para o robô atingir a posição objetivo).
- **TT** é o tempo total de execução da tarefa em segundos (tempo percorrido até o robô retornar à posição inicial).
- **NO** é o número de obstáculos conhecidos no ambiente.
- **Posições** representa as posições centrais dos obstáculos conhecidos no ambiente.
- **VM** representa a velocidade máxima (em milímetros por segundo) que o robô está configurado para alcançar. Um robô móvel possui um limite máximo de velocidade, mas qualquer valor inferior a este limite pode ser configurado pelo usuário para um determinado ambiente.

Esta informação é importante, pois tem influência direta no tempo de realização de uma tarefa.

Tabela 1: Um exemplo do histórico principal do mecanismo

Origem	Objetivo	TP	TT	NO	Posições	VM
(123,664)	(987,1245)	47.1	106.2	5	M1 = (873,232)...	900
(435,344)	(667,-7463)	52.8	120.2	6	C3 = (987,-456)...	900
(123,664)	(987,1245)	46.8	105.6	5	M1 = (858,238)...	900
(435,344)	(667,-7463)	53.4	121.5	6	C3 = (972,-474)...	900
(123,664)	(987,1245)	58.8	124.2	5	M1 = (865,232)...	500
...	...	...	...	...	...	...

4.3.2 Fórmula de previsão

Para o cálculo da previsão, alguns registros são obtidos do histórico principal imediatamente antes do início da tarefa. Conforme mencionado anteriormente, o histórico principal armazena informações sobre as tarefas nas quais o robô não encontrou objetos desconhecidos ao longo da sua trajetória. Uma busca é realizada no histórico e a mesma retorna um conjunto de registros correspondentes a execuções anteriores de tarefas similares à tarefa atual. Na posse destes registros, o algoritmo calcula os tempos médios de execução (parcial e final). O tempo médio parcial corresponde ao tempo médio gasto até o robô atingir a metade da trajetória (posição objetivo) e o tempo médio final corresponde ao tempo médio gasto até a conclusão da tarefa (retorno à posição inicial). Isso mostra que o tamanho do histórico principal tem direta influência no resultado da previsão, pois quanto maior for o conjunto de registros retornado pela busca, maior é a precisão dos tempos médios calculados. A influência do tamanho do histórico principal é mostrada no próximo capítulo desta dissertação.

Durante a realização de uma tarefa, alguns dados são constantemente monitorados, como por exemplo, o tempo de execução e a velocidade do robô. No momento da previsão, o mecanismo assume que os mesmos obstáculos fixos desconhecidos encontrados ao longo da trajetória até a posição objetivo também serão encontrados durante o retorno à posição inicial. De outra forma, o mecanismo assume que os obstáculos móveis desconhecidos encontrados durante a ida não serão encontrados na volta à posição inicial. Porém, uma pequena folga de tempo é reservada no caso de alguns desses obstáculos

móveis ainda se encontrarem no caminho do robô durante a volta. O mecanismo utiliza o módulo de detecção de obstáculos para determinar se os obstáculos desconhecidos encontrados durante a realização da tarefa são fixos ou móveis.

Assim que o robô atingir a posição objetivo e antes de voltar à posição inicial, o mecanismo faz a previsão de perda de *deadline* para a tarefa atual. Para realizar o cálculo, o algoritmo utiliza o tempo parcial de execução até o momento, os tempos médios calculados a partir do histórico e o tempo estimado para o robô se desviar dos obstáculos fixos encontrados ao longo da trajetória. A Eq. (4.1) mostra o cálculo da previsão. "DL" é o *deadline* da tarefa, "TEP" é o tempo de execução parcial até o momento da chegada na posição objetivo, "TMF" e "TMP" são os tempos médios final e parcial calculados a partir do histórico antes do início da tarefa, respectivamente. "TDOF" é o tempo estimado que o robô levará para se desviar dos obstáculos fixos desconhecidos durante a volta à posição inicial e "FDOM" é a folga de tempo reservada no caso dos obstáculos móveis encontrados na ida ainda estiverem presentes na volta. Opcionalmente, antes do início da tarefa, uma previsão inicial de perda de deadline pode ser realizada comparando o deadline da tarefa com o TMF (tempo médio para a realização de tarefas similares sem a presença de obstáculos desconhecidos). Caso o deadline seja menor do que o tempo médio calculado a partir do histórico, muito provavelmente a tarefa não será concluída dentro do prazo estabelecido.

$$DL - TEP \geq TMF - TMP + TDOF + FDOM \tag{4.1}$$

O objetivo da equação de previsão é verificar se existe tempo suficiente para o robô voltar à posição inicial da tarefa sem ultrapassar o *deadline* estabelecido, baseando a decisão nas execuções anteriores de tarefas similares. O lado esquerdo da equação calcula o tempo restante do *deadline*, fazendo a diferença entre o *deadline* e o tempo de execução parcial. O lado direito da equação calcula o tempo estimado para o robô voltar da posição objetivo até a posição inicial. Para calcular esta estimativa, o mecanismo primeiro calcula o tempo médio que o robô leva para voltar da posição objetivo até a posição inicial sem encontrar obstáculos desconhecidos pelo caminho $(TMF - TMP)$, levando em consideração as execuções das tarefas anteriores armazenadas no histórico. O mecanismo considera que o tempo gasto pelo robô no desvio de obstáculos fixos desconhecidos durante a ida até a posição objetivo será aproximadamente o mesmo que o tempo necessário para o desvio destes mesmos obstáculos durante a volta do robô. Este tempo estimado para o desvio de obstáculos fixos tem valor igual a zero caso não seja detectado nenhum obstáculo fixo durante a trajetória do robô até a posição objetivo. Caso contrário, o valor é calculado fazendo-se a diferença entre o tempo de

execução parcial, o tempo médio parcial (o tempo médio que o robô demora para atingir a posição objetivo sem encontrar obstáculos desconhecidos) e o tempo de desvio de obstáculos móveis. Tipicamente, ao ser detectado um obstáculo móvel ao longo da trajetória, um robô móvel diminui a sua velocidade ou até pára em casos extremos, até que o obstáculo saia da sua trajetória. O módulo de detecção de obstáculos mantêm um temporizador que é acionado sempre que um obstáculo móvel é detectado na trajetória e o robô é obrigado a diminuir a velocidade ou parar. O temporizador é pausado quando o obstáculo deixa a trajetória e o robô volta a navegar com o caminho livre. O valor final deste temporizador na chegada do robô à posição objetivo, constitui o valor de TDOM (tempo de desvio de obstáculos móveis). Este valor também é utilizado para calcular o TDOF conforme mostra a Eq. (4.2). O resultado obtido é adicionado ao resultado parcial da fórmula de previsão.

$$TDOF = TEP - TMP - TDOM \tag{4.2}$$

Finalmente, o algoritmo adiciona ao resultado parcial a folga reservada para compensar a presença de obstáculos móveis durante a volta do robô à posição inicial. A folga de tempo reservada para cada obstáculo móvel pode ter um valor fixo e predeterminado ou pode ser definido em tempo de execução.

”FDOM” é a folga total e o seu cálculo é mostrado na Eq. (4.3). ”NOM” representa o número de obstáculos móveis encontrados pelo robô durante a trajetória para a posição objetivo. Ao adicionar o valor da folga total ao resultado parcial anterior, o mecanismo obtém a estimativa do tempo de volta do robô da posição destino até a posição inicial (lado direito da Eq. (4.1)). Se o restante do *deadline* calculado inicialmente (lado esquerdo da equação) for maior ou igual à estimativa do tempo de volta do robô, o mecanismo conclui que provavelmente a tarefa será concluída dentro do prazo estabelecido, ou seja, não há perda de *deadline*. Caso contrário, o *deadline* provavelmente será perdido e ações corretivas podem ser executadas pela aplicação para evitar ou minimizar possíveis prejuízos.

$$FDOM = NOM \times Folga \tag{4.3}$$

Vale notar que se o robô atingir a posição destino sem encontrar nenhum obstáculo desconhecido no trajeto, ”TDOF” e ”FDOM” terão valor igual a zero e a fórmula de previsão é simplificada, conforme mostra a Eq. (4.4)

$$DL - TEP \geq TMF - TMP \tag{4.4}$$

4.3.3 Compensação de mudanças repentinas no ambiente

Conforme mencionado anteriormente, o mecanismo de previsão assume que os mesmos obstáculos fixos encontrados pelo robô durante a sua trajetória até a posição objetivo, serão encontrados novamente na volta, ao contrário dos obstáculos móveis. Como as tarefas do robô geralmente não são muito longas e duram um período de tempo relativamente curto, este cenário é o mais provável de acontecer. Porém, em alguns casos, podem acontecer mudanças repentinas no ambiente e o robô pode encontrar um número diferente de obstáculos fixos na volta à posição inicial, o que pode implicar em algumas previsões erradas por parte do mecanismo. O módulo de compensação é responsável por compensar estas possíveis mudanças no ambiente e melhorar a qualidade das previsões. Este módulo utiliza um histórico adicional, diferente do histórico principal, para armazenar algumas informações sobre tarefas anteriores. Cada ambiente conhecido pelo robô terá um histórico de compensação correspondente, que pode ser consultado para compensar mudanças inesperadas e repentinas no ambiente e melhorar a qualidade das previsões.

O histórico de compensação armazena algumas informações no final de cada tarefa em que o robô tenha encontrado obstáculos desconhecidos durante a sua trajetória. A Tabela 2 mostra a estrutura do referido histórico.

Tabela 2: Um exemplo do histórico de compensação

Data/Hora	Origem	Objetivo	VM	DifObst	DifTempo
Seg 10 Mar 09:04	(425, -8778)	(6255, 766)	900	+1	+7,2 s
Seg 10 Mar 19:23	(-234, 678)	(6255, 766)	900	-1	-9,3 s
Seg 10 Mar 22:04	(425, -8778)	(6255, 766)	900	0	+0,3 s
Seg 10 Mar 22:14	(425, -8778)	(6255, 766)	900	+1	+8,6 s
Ter 11 Mar 09:02	(-234, 678)	(-444, 234)	900	0	-0,4 s
Ter 11 Mar 12:15	(425, -8778)	(6255, 766)	900	+1	+8,4 s
Qua 12 Mar 23:04	(425, -8778)	(-444, 234)	900	+2	+15,3 s
...	...	...	...	...	...

”Data/Hora” representa a data e a hora em que a tarefa foi iniciada. ”Origem” é a posição inicial do robô, ”Objetivo” é a posição objetivo e ”VM” é a velocidade máxima do robô para o ambiente, definida pelo usuário. ”DifObst” é a diferença entre o número de obstáculos fixos desconhecidos encontrados pelo robô durante a volta até a posição origem e o número de obstáculos fixos encontrados durante a sua ida até a posição objetivo. O valor de ”DifObst” é zero se foi encontrado o mesmo número de obstáculos fixos na ida e na volta, $+k$ se o robô encontrou k obstáculos a mais na volta em relação à ida e $-k$ se o robô encontrou k obstáculos a menos durante a volta à posição inicial. O cálculo é mostrado na Eq. (4.5). ”NOF” representa o número de obstáculos fixos desconhecidos encontrados.

$$DifObst = NOF_{volta} - NOF_{ida} \tag{4.5}$$

”DifTempo” é a diferença entre o tempo real que o robô levou para voltar da posição objetivo até a posição inicial e o tempo estimado pelo mecanismo para este percurso (lado direito da equação de previsão), conforme mostra a Eq. (4.6). ”TET” é o tempo de execução total da tarefa (ida e volta) e ”TEP” é o tempo de execução parcial, ou seja, o tempo de ida do robô até a posição objetivo. A diferença entre TET e TEP representa o tempo real que o robô levou para voltar da posição objetivo até a posição de origem.

$$DifTempo = (TET - TEP) - (TMF - TMP + TDOF + FDOM) \tag{4.6}$$

O valor de DifTempo pode ser positivo ou negativo, dependendo se o robô demorou mais ou menos tempo do que o estimado, respectivamente. A ideia é armazenar a diferença entre o tempo real de navegação do robô e a estimativa feita, causada pelos obstáculos encontrados durante a volta até a posição inicial. O histórico de compensação é consultado e ”DifTempo” pode ser utilizado para atualizar a estimativa do tempo de navegação do robô na volta à posição inicial e esse novo valor é então usado no momento da previsão.

O histórico de compensação pode ser consultado utilizando diferentes critérios de busca. O mecanismo primeiro seleciona o histórico correspondente ao ambiente em que a tarefa está sendo realizada e executa uma busca considerando apenas os registros correspondentes a tarefas similares (com a mesma posição inicial e a mesma posição objetivo). Os critérios de busca podem ser os seguintes:

- **Últimas N tarefas realizadas:** O mecanismo busca pelas N últimas execuções de tarefas similares à tarefa sendo realizada. A ideia é que o ambiente provavelmente permanece similar ao que era nas últimas

execuções da tarefa. Se $N = 10$ por exemplo, o mecanismo seleciona as últimas 10 execuções de tarefas similares. Se $N = 1$, o mecanismo busca apenas o registro da última execução de uma tarefa similar.

- **Todas as tarefas realizadas:** Todos os registros referentes a tarefas similares à tarefa atual são extraídos do histórico de compensação.

- **Tarefas realizadas num determinado intervalo de tempo:** O mecanismo seleciona apenas registros de execuções específicas da tarefa, considerando a data e/ou hora em que as tarefas foram iniciadas. Em uma fábrica ou armazém, existem certos momentos de pico em que o ambiente se altera com frequência, assim como momentos de pouco ou nenhum movimento. Esta informação é importante pois pode influenciar diretamente no resultado das previsões de perda de *deadline*. O mecanismo pode selecionar apenas registros de tarefas que foram realizadas num determinado intervalo de tempo. O filtro pode ser ainda mais restrito e considerar apenas execuções de tarefas realizadas num intervalo de tempo em um determinado dia de semana, ou ainda num mês específico, caso o mês também seja relevante.

A consulta ao histórico de compensação retorna um conjunto de registros com informações relevantes, referentes a execuções anteriores de tarefas similares à tarefa atual. Estes registros são utilizados para se obter a média do DifTempo e atualizar o valor do tempo estimado para o desvio de obstáculos fixos desconhecidos durante a volta do robô ("TDOF" na Eq. (4.1)). Antes do cálculo da média, o mecanismo verifica se a maioria dos registros possui o mesmo valor de DifObst. Caso isso aconteça, os registros pertencentes a esta maioria são considerados no cálculo da média do DifTempo. Caso contrário, todos os registros retornados pela busca são considerados. Se o critério de busca utilizado for o critério das N últimas tarefas executadas e não for encontrado um padrão de maioria nos registros retornados, a busca é estendida por mais N registros, até que a maioria seja encontrada. De posse desta maioria, esses registros são utilizados para o cálculo da média do DifTempo e esta média é então utilizada para atualizar o valor de TDOF. A média calculada pode assumir um valor negativo ou positivo, dependendo se em média o tempo real de volta do robô à posição inicial é menor ou maior do que o tempo estimado pelo mecanismo, respectivamente. Tendo o conjunto de registros para o cálculo da média do DifTempo, e seja n o tamanho desse conjunto, o valor de TDOF é atualizado conforme mostra a Eq. (4.7)

$$TDOF_{novo} = TDOF_{velho} + \frac{1}{n}\sum_{i=1}^{n} DifTempo_i \tag{4.7}$$

No exemplo mostrado na Tabela 2, se na tarefa atualmente em execução a posição inicial possuir coordenadas $(425, -8778)$ e a posição objetivo possuir coordenadas $(6255, 766)$, usando o primeiro critério de busca com $N = 4$ percebe-se que em 3 das últimas 4 execuções da tarefa o robô encontrou um obstáculo a mais do que o esperado durante a sua volta à posição inicial. Assim sendo, o mecanismo considera que na execução atual o robô provavelmente irá encontrar o mesmo cenário e adiciona 8.06 segundos a TDOF $((7.2s + 8.6s + 8.4s)/3)$.

Após um determinado número de previsões consecutivas incorretas, a compensação é habilitada automaticamente e a mesma é desabilitada quando o mecanismo acertar o resultado da previsão. A Figura 6 sumariza o mecanismo proposto, apresentando um diagrama de fluxo do processo de previsão. O diagrama apresenta o fluxo de execução do mecanismo, desde o início de uma determinada tarefa de navegação, até o cálculo do resultado da previsão. Também é mostrada a interação do mecanismo de previsão com o mecanismo de compensação, que é habilitado automaticamente sempre que necessário.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo abordou a proposta desta dissertação, apresentando o modelo de tarefas, a arquitetura do sistema e o mecanismo de previsão de perda de *deadline*. O mecanismo pode ser aplicado em qualquer robô móvel que utilize uma arquitetura híbrida e não depende de nenhuma implementação específica dos sistemas de navegação, planejamento de trajetória e detecção de obstáculos.

Obstáculos móveis inesperados encontrados pelo robô durante a volta para a posição inicial tendem a não ter grande influência no resultado da previsão de perda de *deadline*, pois em geral, o tempo que o robô leva para evitar esses obstáculos é consideravelmente menor do que no caso de obstáculos fixos. O tamanho do obstáculo móvel e a sua velocidade influenciam diretamente no tempo necessário para o desvio por parte do robô.

O próximo capítulo aborda a validação do sistema proposto, apresentando as simulações e testes realizados, os hardwares e softwares utilizados e os resultados obtidos.

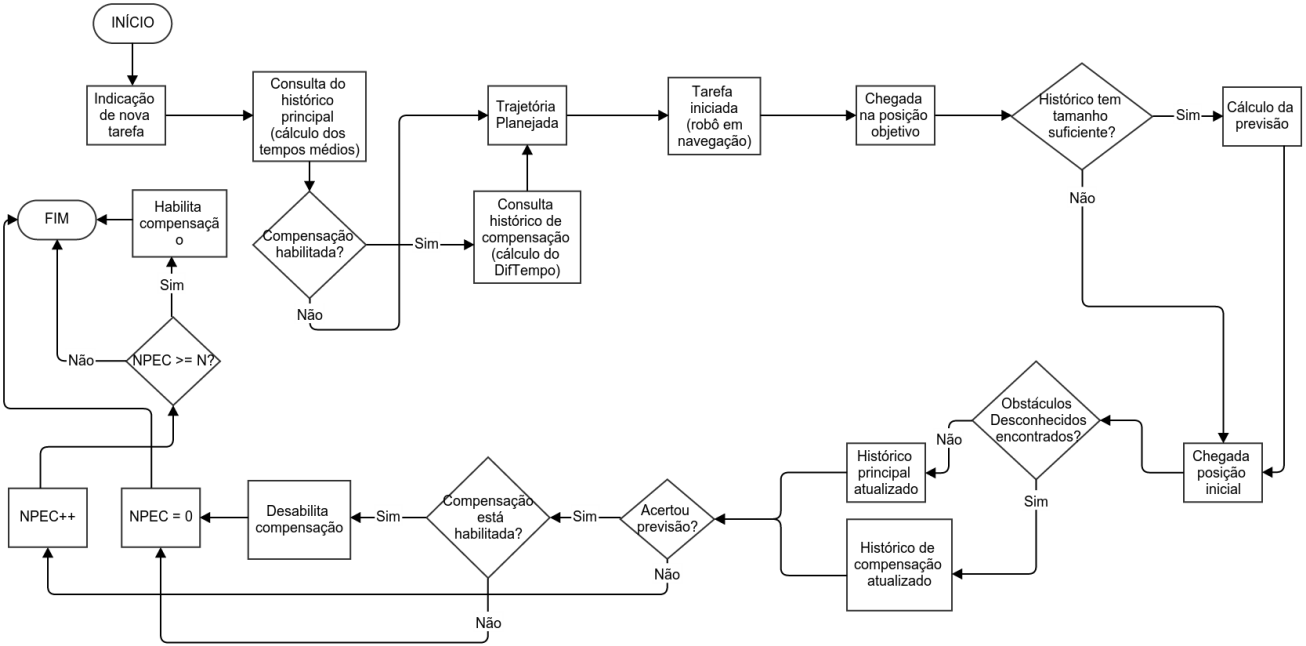

Figura 6: Diagrama de fluxo do processo de previsão
NPEC = Número de previsões erradas consecutivas.
N = Número máximo de previsões erradas consecutivas antes de habilitar a compensação

5 SIMULAÇÕES, TESTES E RESULTADOS

Este capítulo visa validar o mecanismo proposto. São descritos o robô e os sensores utilizados nos testes, bem como os softwares utilizados para implementar e testar o mecanismo. Foram utilizados os sistemas de localização, planejamento de trajetória e navegação desenvolvidos pela fabricante do robô e estes se mostraram eficientes durante as simulações e testes. Esses sistemas são proprietários e de código fechado, sendo que as suas implementações são mantidas em segredo comercial. Foi necessário implementar o módulo de detecção de obstáculos para se ter um controle maior na detecção de obstáculos desconhecidos.

Várias simulações e testes foram realizados em ambientes simulados e reais e os resultados são apresentados neste capítulo. O capítulo se encerra comparando o mecanismo desenvolvido com um mecanismo mais simples que não usa histórico de informações.

5.1 HARDWARE UTILIZADO

Durante o desenvolvimento do mecanismo proposto, vários experimentos foram realizados utilizando o robô *Pioneer 3-DX* (ou simplesmente *P3-DX*) e seus principais sensores acoplados. O Pionner 3-DX é um robô que utiliza a arquitetura híbrida sendo que um mapa é utilizado para representar o ambiente conhecido (representação métrica). O laser *Sick LMS-200* é utilizado na geração do mapa do ambiente, localização do robô e planejamento de trajetória. Para o robô se locomover de uma posição até outra dentro do ambiente, inicialmente é feito um planejamento global de trajetória. O sistema de navegação tenta garantir que o robô siga essa trajetória gerada de forma mais fiel possível. Durante a navegação, caso seja detectado algum obstáculo desconhecido ao longo da trajetória do robô, um novo planejamento local é realizado, de forma que o robô possa desviar do obstáculo e volte à trajetória planejada inicialmente. Caso isto não seja possível, ou seja, caso a trajetória estiver totalmente bloqueada, um novo planejamento global é feito e uma nova trajetória é definida. O mapa do ambiente pode ser criado manualmente pelo usuário ou gerado automaticamente através de leituras realizadas pelo sensor laser enquanto o robô se movimenta pelo ambiente.

5.1.1 Robô Pioneer 3-DX

Pioneer é uma família de robôs, com tração em duas ou quatro rodas desenvolvida pela empresa *MobileRobots*. Os robôs mais novos desta família são o *Pioneer 3-DX* e o *Pioneer* 3-AT. Estas plataformas de pesquisa e desenvolvimento compartilham uma arquitetura comum e software com todas as outras plataformas da empresa, incluindo os robôs móveis *AmigoBot, PeopleBot V1, Performance PeopleBot e PowerBot* (MOBILEROBOTS, 2013f).

O *Pioneer 3-DX* (Figura 7), é um robô móvel utilizado para pesquisa e experimentos na área de robótica móvel. Tem-se tornado cada vez mais uma plataforma de referência para implementar e comparar diferentes algoritmos. O P3-DX possui um corpo rígido de alumínio e oferece um computador embarcado (com sistema operacional Linux) que possui uma série de componentes, onde um usuário pode programá-lo de acordo com sua necessidade de aplicação. Dentre esses componentes podem ser citados recursos para processamento de visão, comunicação via *Ethernet* (com e sem fios), dezesseis sonares (sendo oito atrás e oito à frente) para detecção de objetos localizados entre 15 centímetros e 7 metros de distância, quatro grandes rodas acopladas a motores potentes capazes de produzir uma velocidade máxima de 1,6 metros/segundo e carregar um peso de até 23 kg, pára-choques com sensores de toque e 3 baterias que garantem autonomia de 8 a 10 horas (MOBILEROBOTS, 2013f).

Figura 7: O robô Pionner 3-DX
Fonte: MobileRobots

A principal vantagem do Pioneer 3-DX é a sua confiabilidade e durabilidade, tornando-se adequado para diferentes tipos de testes e experimentos de pesquisa. Ele é totalmente programável e oferece grande controle sobre

diferentes características que podem ser de interesse para os pesquisadores. Além disso, existe uma grande variedade de acessórios e sensores que podem ser acoplados, aumentando a versatilidade do robô. Um dos sensores utilizados e fundamental para a localização precisa do robô, a geração do mapa do ambiente e planejamento da trajetória é o laser *LMS-200* descrito na próxima seção (MOBILEROBOTS, 2013c).

5.1.2 Laser Sick LMS-200

O LMS-200 (Figura 8) é um sensor laser fabricado pela empresa *Sick Sensor Inteligence* e que pode ser adquirido como acessório opcional do Pionner 3-DX. Um sensor laser é fundamental para a correta localização do robô, uma vez que a localização por sonares é bastante imprecisa (MOBILEROBOTS, 2013c).

Figura 8: O Laser Sick LMS-200
Fonte: MobileRobots

O Sick LMS-200 foi projetado para uso em ambientes internos e possui um ângulo de ação de até 190 graus com uma frequência de leitura de até 75 Hz. O laser tem um alcance de até 80 metros, pesando pouco mais de 3 kg. A Figura 9 mostra o robô Pioneer 3-DX equipado com o laser.

Figura 9: O laser LMS-200 acoplado ao Pioneer 3-DX
Fonte: MobileRobots

5.2 SOFTWARE UTILIZADO

O mecanismo de previsão foi implementado utilizando a linguagem C++. Existem diferentes plataformas de software utilizadas neste trabalho. As plataformas são utilizadas para controlar o robô Pioneer 3-DX e os sensores acoplados. A principal plataforma utilizada para controlar e programar o robô é o *ARIA*. O ARIA oferece várias classes e funções para várias operações na plataforma robótica e oferece comandos simples para controlar várias manobras complexas do robô. Outro software utilizado é o *ARNL*, que é uma extensão do ARIA para localização, navegação, planejamento de trajetória e comunicação através da rede.

O simulador do Pioneer também é utilizado para testar diferentes algoritmos e exemplos de programas antes de serem implementadas no robô real. O *Mapper3* é utilizado para gerar o mapa do ambiente manualmente ou a partir das leituras do laser enquanto o robô é tele-operado. Por fim, a plataforma *MobileEyes* fornece uma interface gráfica fácil de usar para o planejamento de trajetória e tarefas de localização. A seguir é apresentada uma breve descrição de cada um desses softwares.

5.2.1 ARIA

ARIA (*Advanced Robotics Interface for Applications*) é um ambiente de desenvolvimento de código livre, orientado a objetos e baseado em C++ para controle de robôs móveis inteligentes incluindo o microcontrolador do

robô e sistemas acessórios. A programação é feita utilizando a linguagem C++, mas recentemente foi introduzido o suporte às linguagens Java e Python (MOBILEROBOTS, 2013a). O ARIA é multi-plataforma (Windows e Linux) e facilita o gerenciamento e controle de acesso ao robô. Ele também fornece controle eficaz e eficiente utilização de vários sensores do robô (MOBILEROBOTS, 2013a).

O ARIA também fornece várias ferramentas úteis para programação do robô em geral. Além de fornecer uma API (*Application Programming Interface*) completa para controle do robô e dos seus sensores, o ARIA também serve como base para outras bibliotecas, fornecendo recursos adicionais. Para a criação de aplicações com rotinas de navegação avançada estão disponíveis as bibliotecas adicionais ARNL e SONARNL. Para se comunicar com o MobileEyes ou para comunicação em geral através da rede pode-se usar a biblioteca ArNetworking.

O ARIA oferece ampla gama de classes para lidar com o robô móvel de forma eficiente e confiável, na realização de uma tarefa específica. A documentação do ARIA é vasta e uma importante fonte de consulta, pois descreve todas as classes em detalhe, incluindo suas funções, variáveis e formas de uso.

5.2.2 ARNL

ARNL é um conjunto de pacotes de software construídos em cima do ARIA para navegação inteligente e localização dentro de ambientes fechados que foram mapeados. ARNL é uma solução completa e autônoma de localização e navegação, bem como um conjunto abrangente de ferramentas de desenvolvimento e exemplos de aplicações que permitem a execução de tarefas avançadas utilizando o sensor laser *Sick*.

A localização permite ao programa manter o controle de onde o robô se encontra e a navegação permite ao robô chegar a um determinado destino. As bibliotecas do ARNL permitem executar e coordenar as tarefas de localização e navegação, corrigindo automaticamente a posição do robô e o dirigindo até o objetivo especificado no programa de controle. O ARNL faz uso dos dados obtidos pelo laser para melhor precisão na localização e navegação (MOBILEROBOTS, 2013b).

5.2.3 MobileSim

MobileSim é um software para simulação de robôs móveis e seus ambientes, realização de debug e experimentos em conjunto com programas baseados no ARIA.

O software MobileSim converte um mapa (um arquivo .map contendo a descrição do ambiente) para o ambiente interno do simulador, inserindo um modelo de robô simulado. Também provê uma conexão com o Pioneer simulado via porta TCP 8101 (similar à conexão via porta serial com o Pioneer real). A maioria dos programas baseados no ARIA conecta automaticamente na porta TCP caso ela esteja disponível (MOBILEROBOTS, 2013e).

Quando um programa baseado em ARIA é executado, o mesmo tenta se conectar ao robô real através da porta TCP, mas caso o robô não seja encontrado, o programa tenta se conectar a um simulador MobileSim através da porta TCP. Assim, o MobileSim não pode ser acessado, na presença de um robô real. Além do modelo de robô, vários sensores como lasers e sonares também são modelados pelo MobileSim, o que permite simular o funcionamento destes dispositivos.

5.2.4 Mapper3

Mapper3 é um software que fornece uma interface gráfica para criação e edição de mapas de ambientes onde os robôs móveis irão se locomover. O mapa pode ser criado manualmente pelo usuário ou através dos dados obtidos pelo sensor laser. Neste último caso, o robô é inicialmente tele-operado por todo o ambiente enquanto o laser captura informações. Em seguida, o Mapper3 pode usar o arquivo gerado pelo laser para gerar o mapa automaticamente, refazendo a trajetória do robô e comparando com os dados lidos pelo laser. Mapas criados pelos Mapper3 são usados pelo ARNL, MobileSim e outras plataformas. A Figura 10 mostra a interface gráfica do Mapper3 (MOBILEROBOTS, 2013d).

5.2.5 MobileEyes

MobileEyes é uma interface gráfica do usuário para controlar o robô remotamente. MobileEyes comunica com o servidor via rede TCP/IP e, uma vez conectado, o robô envia ao MobileEyes uma cópia do mapa usado para navegar no ambiente. O mapa e a localização do robô dentro deste mapa, bem como os objetivos e outras informações operacionais são exibidas na janela

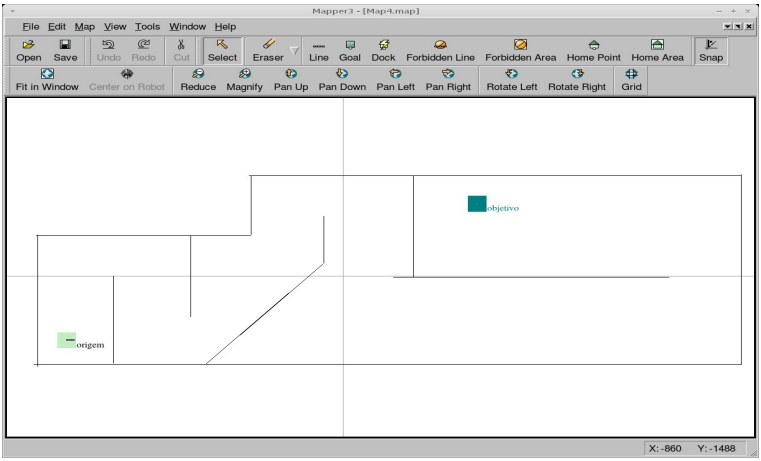

Figura 10: Interface gráfica do Mapper3

do MobileEyes. As funcionalidades do MobileEyes incluem definir objetivos, navegar o robô até algum objetivo, mostrar o trajeto gerado pelo processo de planejamento de trajetória, etc. A interface gráfica do MobileEyes pode ser vista na Figura 11.

5.3 MÓDULO DE DETECÇÃO DE OBSTÁCULOS

A Figura 5 apresentou a arquitetura geral do sistema proposto. Na implementação do referido sistema foram utilizados os sistemas de localização, navegação e planejamento de trajetória disponibilizados pela fabricante do robô que foi utilizado nos testes. No entanto, foi necessário implementar o módulo de detecção de obstáculos, pois o mecanismo de detecção de obstáculos oferecido pela fabricante é fechado e interno ao sistema de navegação, não oferecendo o controle necessário para a implementação do mecanismo de previsão de perda de *deadline*.

Foi implementado um sistema que utiliza as leituras do sensor laser para detectar a presença de obstáculos inesperados. O sistema implementado primeiro obtém a trajetória (conjunto de posições no ambiente por onde o robô vai passar) através do módulo de planejamento de trajetórias. Uma vez obtidos os pontos da trajetória, o algoritmo traça segmentos de reta imaginários entre cada par de pontos consecutivos da trajetória. Em seguida o algoritmo faz uso dos recursos avançados oferecidos pelo ARNL para manipulação de lasers e procura por padrões de segmentos de retas nos va-

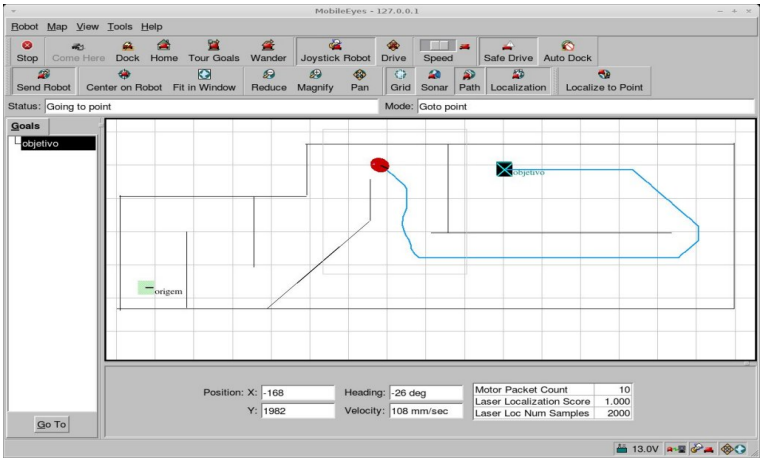

Figura 11: Interface gráfica do MobileEyes

lores obtidos na leitura do laser, ou seja, os obstáculos são representados por um conjunto de segmentos de retas. Por fim, o algoritmo procura por pontos de interseção entre os segmentos de reta que representam os obstáculos e os segmentos que representam a trajetória. Caso seja encontrada uma interseção ou um dos pontos extremos do segmento de reta que representa o obstáculo esteja a uma distância da trajetória insuficiente para o robô passar sem alterar a sua trajetória, é feita a detecção de um obstáculo. Este obstáculo é monitorado por um curto período de tempo enquanto o robô se aproxima do mesmo. Se no final deste período de tempo for detectada uma mudança na posição do segmento de reta que representa o obstáculo no ambiente, o obstáculo é considerado como sendo móvel. Caso contrário, ou seja, caso o obstáculo não mudar de posição durante o período de tempo estipulado, o mesmo é considerado como fixo.

A Figura 12 ilustra o funcionamento do módulo de detecção de obstáculos. Os pontos em azul representam posições por onde o robô deve passar (a trajetória). As linhas em azul são obstáculos representados por segmentos de reta após leitura dos valores retornados pelo laser. Os pontos em vermelho representam a posição na trajetória onde será necessário um desvio do robô, ou seja, onde um obstáculo obstrui o caminho do robô.

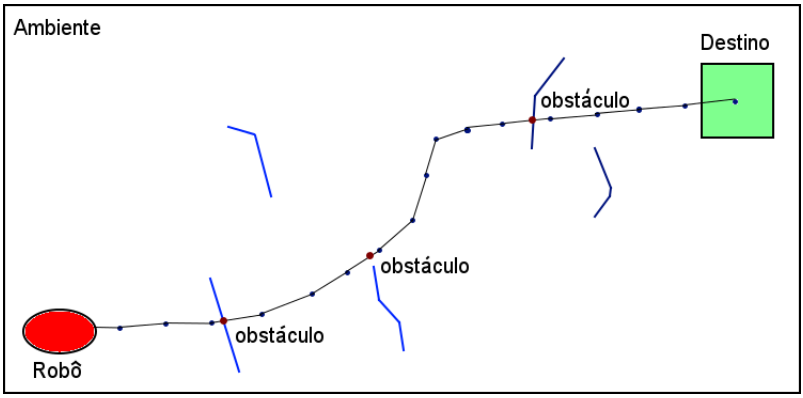

Figura 12: Funcionamento do módulo de detecção implementado

5.4 SIMULAÇÕES E TESTES REALIZADOS

Para validar o mecanismo, simulações e testes foram realizados em diferentes condições, com diferentes disposições de obstáculos no ambiente e históricos com tamanhos diferentes. Primeiramente as simulações foram realizadas em ambientes estáticos e simulados, utilizando o MobileSim para simular o comportamento do robô Pionner 3-DX. Após confirmar o correto funcionamento do mecanismo nos ambientes simulados, testes foram realizados em um ambiente real utilizando o próprio robô. A métrica utilizada para avaliar a qualidade das previsões é a taxa de previsões corretas, que indica a porcentagem de acertos nas previsões do mecanismo.

5.4.1 Ambiente simulado

Neste caso, o ambiente simulado é estático, ou seja, contém apenas obstáculos fixos e conhecidos. Foram utilizadas três configurações diferentes do ambiente, cada uma com cinco obstáculos em diferentes posições. Em cada configuração do ambiente, foram realizadas 100 simulações, variando o tamanho do histórico principal do mecanismo. Foram utilizados históricos com 50, 200, 600 e 1000 registros. Para cada tamanho de histórico, foram realizadas 100 simulações, totalizando 400 simulações para cada configuração de ambiente e 1200 simulações no total. Além de mostrar que o mecanismo funciona, a intenção era mostrar que o tamanho do histórico tem influência

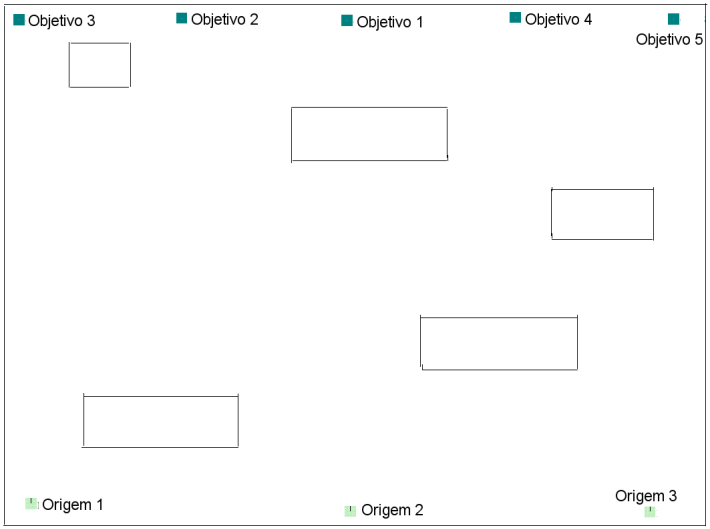

Figura 13: Primeira configuração do ambiente

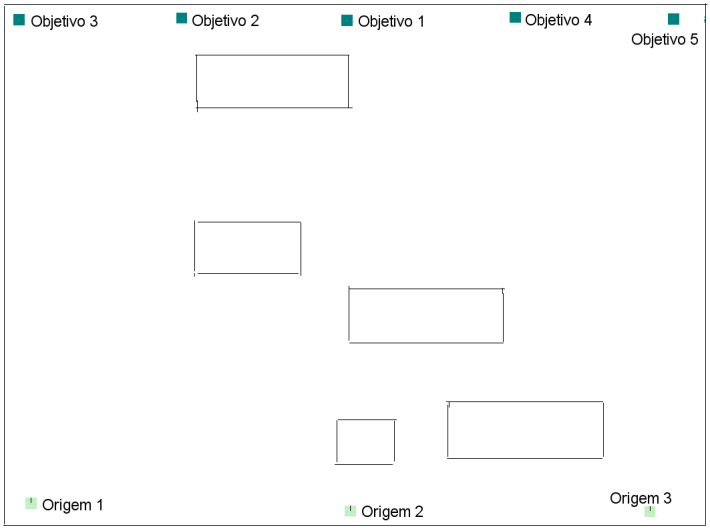

Figura 14: Segunda configuração do ambiente

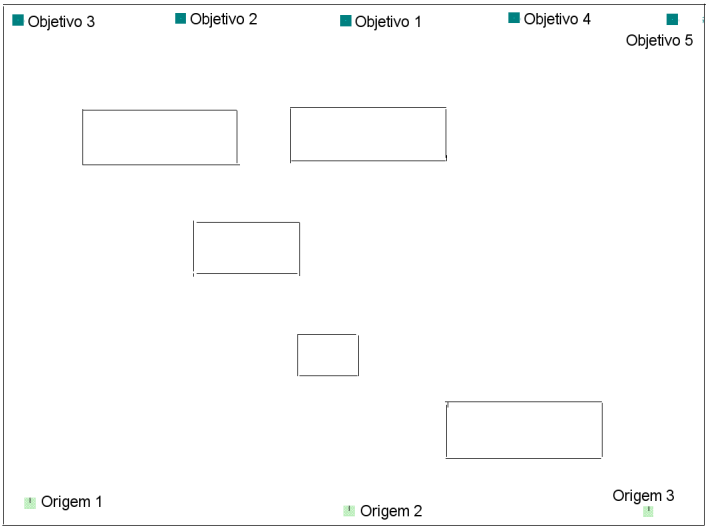

Figura 15: Terceira configuração do ambiente

na qualidade das previsões. A Figura 13, Figura 14 e Figura 15 mostram as três configurações diferentes do ambiente. Inicialmente o mecanismo foi executado em modo de aprendizagem até o histórico principal atingir o tamanho desejado. Em seguida, o modo de previsão foi habilitado e as simulações realizadas.

As simulações foram realizadas utilizando o MobileSim. O *deadline* utilizado nas simulações foi o tempo médio final de execução de tarefas similares, calculado a partir do histórico imediatamente antes do início de uma tarefa. A Figura 16 mostra uma simulação sendo executada no MobileSim.

5.4.1.1 Resultados

As simulações realizadas mostraram que o mecanismo de previsão apresenta bons resultados em ambientes estáticos, é confiável e garante uma taxa de previsões corretas maior do que 90%. A Tabela 3 mostra os tempos mínimos, médios e máximos de execução (em segundos) das tarefas nas simulações com 1000 registros no histórico. C1, C2 e C3 representam as três configurações do ambiente.

Conforme esperado, a qualidade das previsões de perda de *deadline*

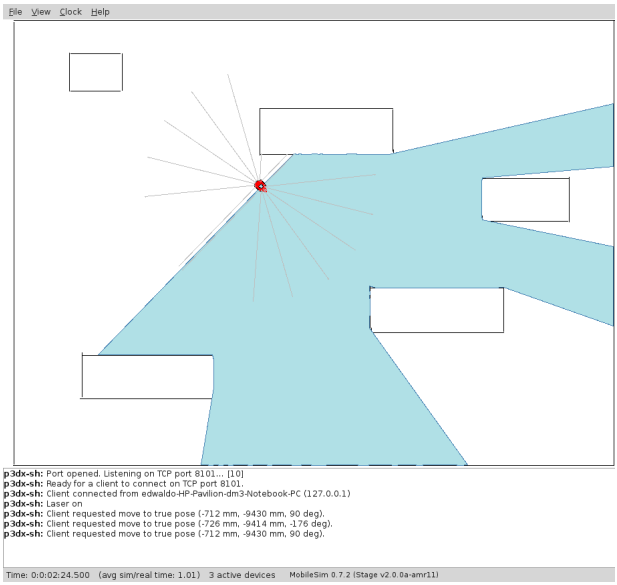

Figura 16: Simulação sendo executada no MobileSim

depende do tamanho do histórico principal do mecanismo. Com apenas 50 registros no histórico, foi obtida uma taxa de previsões corretas variando de 52% a 62%. Com 200 e 600 registros, a taxa variou de 72% a 77% e de 84% a 87%, respectivamente. Finalmente, com 1000 registros no histórico principal, a taxa de previsões corretas ultrapassou os 90%.

A Figura 17 mostra a taxa de previsões corretas de todas as simulações em cada configuração do ambiente. A Figura 18 mostra a influência do tamanho do histórico na taxa de previsões corretas. Como foi alcançada uma taxa superior a 90%, não foi necessário aumentar ainda mais o tamanho do histórico principal, mas isso pode ser feito para melhorar a qualidade das previsões, caso seja necessário. O tamanho ótimo do histórico principal depende do número de ambientes em que as tarefas serão realizadas e do número de tarefas possíveis (quanto maior o número de posições origem e posições objetivos, maior o número de tarefas possíveis). No caso das simulações realizadas, com as 3 configurações diferentes do ambiente e diferentes posições de origem e objetivo, constatou-se que 1000 é o tamanho ótimo. Note-se que o tamanho do histórico não tem influência direta no desempenho da execução das tarefas e no funcionamento do mecanismo, pois a consulta ao histórico é feita antes do início das tarefas.

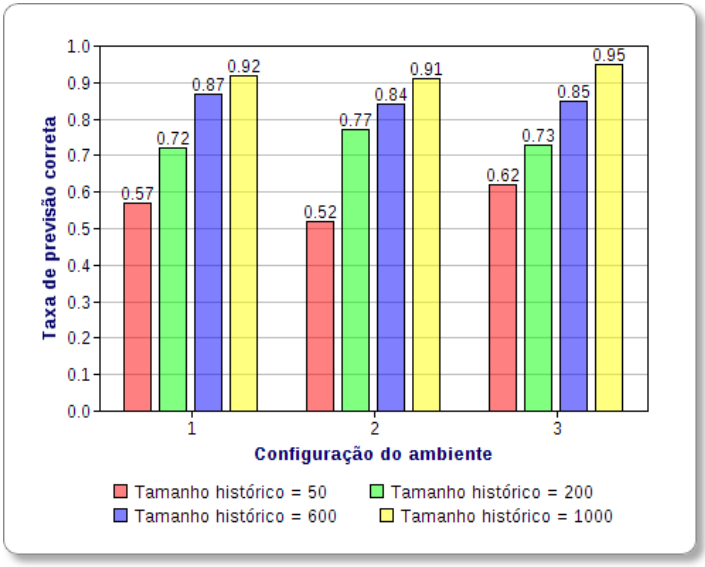

Figura 17: A taxa de previsões corretas nas simulações em ambientes simulados e estáticos

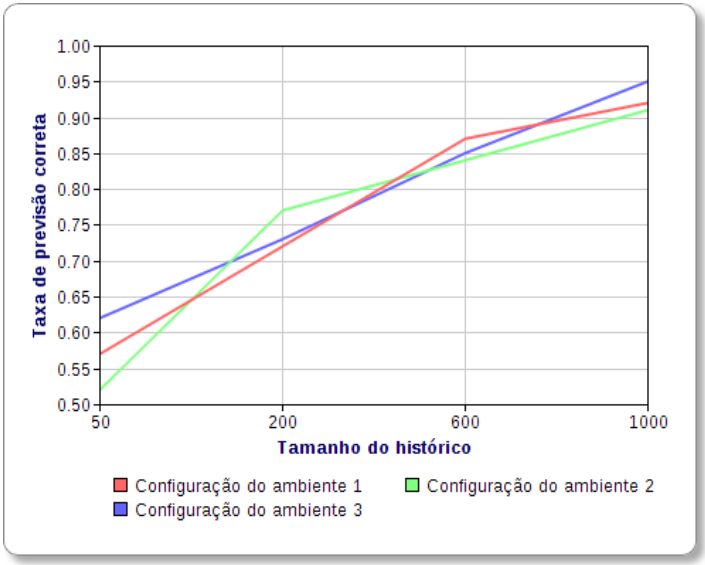

Figura 18: A influência do tamanho do histórico na taxa de previsões corretas

Tabela 3: Resultados da simulação. Tamanho do histórico = 1000

		T Mínimo	T Médio	T Máximo
C1	Tempo parcial de execução	40.97	42.56	45.12
	Tempo total de execução	99.95	102.24	104.73
C2	Tempo parcial de execução	41.22	44.23	45.78
	Tempo total de execução	102.23	104.32	106.23
C3	Tempo parcial de execução	43.83	46.33	48.56
	Tempo total de execução	105.34	108.77	110.97

5.4.2 Ambiente real

Após realizar as simulações nos ambientes estáticos, foi possível confirmar que o mecanismo funciona e é efetivo para estes casos. O próximo passo consistiu em estender os testes para o mundo real, utilizando o robô Pionner 3-DX.

Antes de executar os testes, foi necessário criar o mapa do ambiente desejado. O ambiente escolhido foi o corredor do terceiro andar do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina. Para criar o mapa, o robô Pionner 3-DX foi teleoperado manualmente ao longo do corredor enquanto o laser escaneava o ambiente. Este processo resultou na criação de um arquivo de log, que foi aberto no Mapper3 e utilizado para gerar o mapa do ambiente. A Figura 19 mostra o mapa resultante em que foi definida uma posição inicial (origem) para o robô e duas posições objetivo. A posição objetivo utilizada para realização dos testes é a posição "Objetivo1".

Após a criação do mapa, foi criado um histórico principal para o mecanismo, com tamanho igual a 200. Como o ambiente possui apenas uma posição inicial e duas posições objetivo, um histórico com 200 registros é o suficiente para se obter uma alta taxa de previsões corretas. Em seguida, o mecanismo foi executado em modo de aprendizagem por 200 vezes sem a presença de obstáculos desconhecidos, até o histórico ficar completo.

O *deadline* escolhido para cada execução de tarefa foi gerado em tempo de execução de forma aleatória, no intervalo de tempo $[TMF - 10s, TMF + 30s]$. Conforme mencionado anteriormente, TMF é o tempo médio final de execução calculado a partir do histórico, ou seja, o tempo médio que o robô demora para completar a tarefa sem encontrar nenhum obstáculo desconhecido ao longo da sua trajetória. O tempo de folga para compensar pela

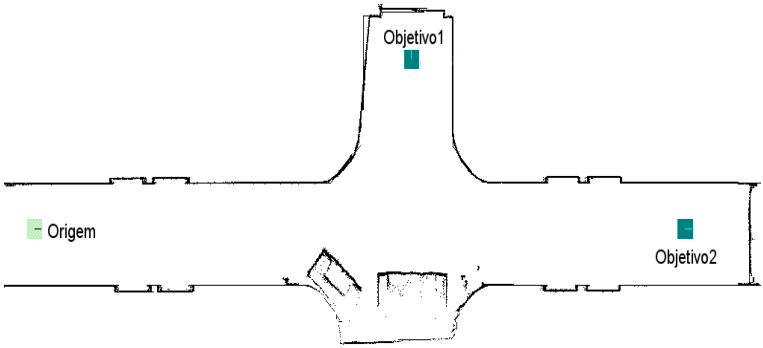

Figura 19: O mapa gerado do ambiente

presença de obstáculos móveis na volta do robô é calculado em tempo de execução como sendo 4 vezes menor do que o tempo médio para o robô desviar de um obstáculo fixo desconhecido. Esse valor foi definido com base na observação dos tempos de desvios de obstáculos móveis e fixos em vários testes realizados.

Os testes no ambiente real foram divididos em 3 cenários distintos, que serão descritos nas próximas seções. Em todos os cenários, o número de obstáculos desconhecidos fixos e móveis varia de 0 a 5. Estes obstáculos consistem em gavetas, caixas e cadeiras, no caso dos obstáculos fixos. Já os obstáculos móveis consistem em pessoas que podem obstruir temporariamente a trajetória do robô.

As Figuras 20 e 21 mostram o robô Pionner 3-DX realizando tarefas e evitando obstáculos desconhecidos na sua trajetória, respectivamente.

5.4.2.1 Cenário 1

Neste cenário, o ambiente não sofre alterações repentinas durante a execução da tarefa, ou seja, o ambiente encontrado pelo robô durante a sua ida até a posição objetivo é o mesmo que o encontrado na volta. Isto significa que os mesmos obstáculos desconhecidos encontrados na ida do robô são encontrados na volta até a posição de origem. Este é o cenário mais provável de acontecer no mundo real caso a duração da tarefa não seja muito longa.

Foram realizados 100 testes neste cenário, variando a quantidade de obstáculos desconhecidos fixos, mas mantendo os mesmos obstáculos durante toda a tarefa. De forma contrária, os obstáculos móveis (pessoas) encontrados pelo robô ao longo da sua ida até a posição objetivo, não permaneceram na

Figura 20: Robô realizando tarefa

Figura 21: Robô evitando obstáculo

trajetória do robô durante a sua volta. Ao final de cada tarefa, foi alterado o número e/ou o posicionamento dos obstáculos fixos ao longo da trajetória do robô. De forma aleatória, obstáculos móveis cruzaram a trajetória do robô apenas durante a sua ida.

5.4.2.1.1 Resultados

No final dos 100 testes, foi verificado o resultado das previsões de perda de *deadline* correspondentes. A taxa de previsões corretas foi de 92%. O gráfico da Figura 22 faz uma comparação entre o tempo (em segundos) que o robô levou para voltar à posição inicial e o tempo que o mecanismo estimou para esta volta no momento da previsão, nas 100 execuções realizadas no cenário 1.

5.4.2.2 Cenário 2

Ao contrário do cenário 1, neste cenário pode haver mudanças bruscas e repentinas no ambiente, ou seja, o número de obstáculos fixos desconhecidos encontrados durante a ida do robô pode variar durante a sua volta até a posição inicial.

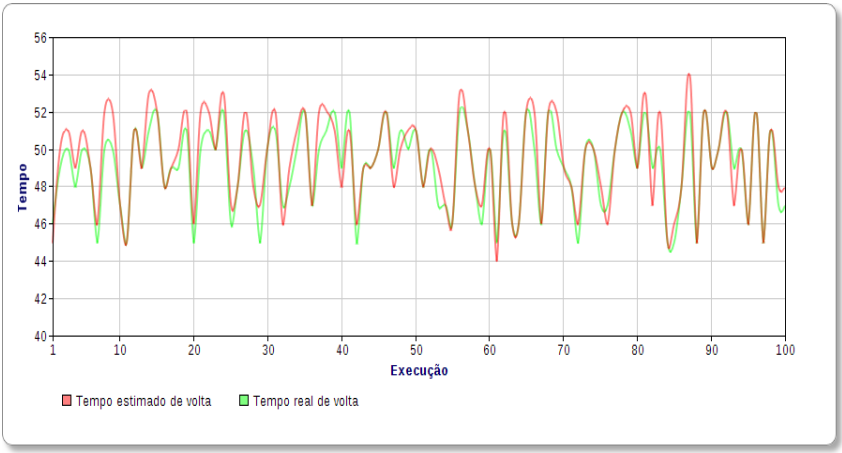

Figura 22: Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 1

Caso o número de obstáculos na volta seja maior ou menor do que o esperado, essa mudança pode causar previsões incorretas de perda de *deadline*, pois o tempo para desvio desses obstáculos adicionais, pode fazer com que o robô perca o *deadline*, mesmo tendo o mecanismo previsto o contrário. A mesma variação de número pode acontecer com os obstáculos móveis. O mecanismo supõe que os obstáculos móveis encontrados na ida, não estarão presentes na volta do robô, mas guarda uma pequena folga de tempo para compensar caso alguns estejam. Alguns desses obstáculos podem realmente estar presentes na trajetória do robô durante a volta, mas no geral tendem a não influenciar muito no resultado da previsão. Neste cenário o módulo de compensação se encontra permanentemente desabilitado. Para identificar a influência no resultado das previsões das mudanças do número de obstáculos desconhecidos, foram realizados mais 100 testes com o módulo de previsão desligado. Estes testes foram realizados alternadamente em grupos de 5 tarefas, sendo 5 execuções de tarefas realizadas sem mudança repentina no ambiente seguidas por 5 execuções com mudanças (aumento do número de obstáculos desconhecidos) e assim alternadamente.

5.4.2.2.1 Resultados

A taxa de previsões corretas caiu drasticamente como esperado e o resultado obtido foi de 59%. Sem o mecanismo de compensação habilitado, o mecanismo falhou várias previsões devido às mudanças repentinas no ambiente no momento da volta do robô à posição inicial. O gráfico da Figura 23 compara o tempo (em segundos) de volta do robô à posição de origem e o valor estimado no momento da previsão, nas 100 execuções realizadas no cenário 2. Devido às mudanças inesperadas no ambiente e a falta do uso do módulo de compensação, a diferença entre os tempos foi considerável.

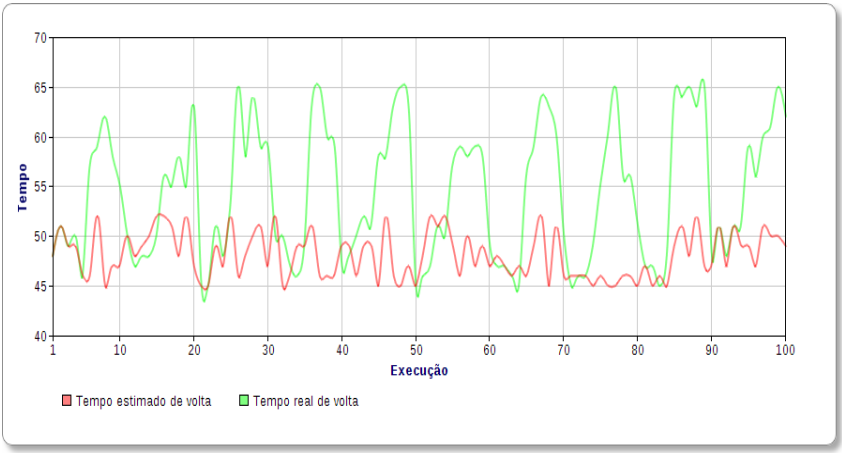

Figura 23: Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 2

Em seguida foram realizados mais 100 testes alterando apenas o número de obstáculos fixos presentes na volta do robô. Em todos os testes realizados, durante a ida até a posição objetivo não foi colocado nenhum obstáculo desconhecido na trajetória do robô. Porém, na volta foi adicionado um obstáculo extra a cada 20 execuções. Isto significa que nas primeiras 20 execuções o robô encontrou 1 obstáculo fixo na volta. Nas próximas 20 execuções encontrou 2 obstáculos e assim sucessivamente até totalizar as 100 execuções. Estes testes visaram determinar a influência da presença de obstáculos fixos inesperados no resultado da previsão. A Figura 24 mostra a taxa de previsões corretas nestes testes. É mostrada a taxa para as situações em que foram encontrados de 1 a 5 obstáculos a mais do que o esperado durante a volta do robô. Como pode ser visto, a influência de obstáculos fixos desconheci-

dos durante a volta do robô no resultado da previsão de perda de *deadline* é grande. Quanto maior for o número de obstáculos inesperados encontrados, maior a probabilidade de o mecanismo errar a previsão pois o tempo de desvio destes obstáculos pode ser relativamente grande.

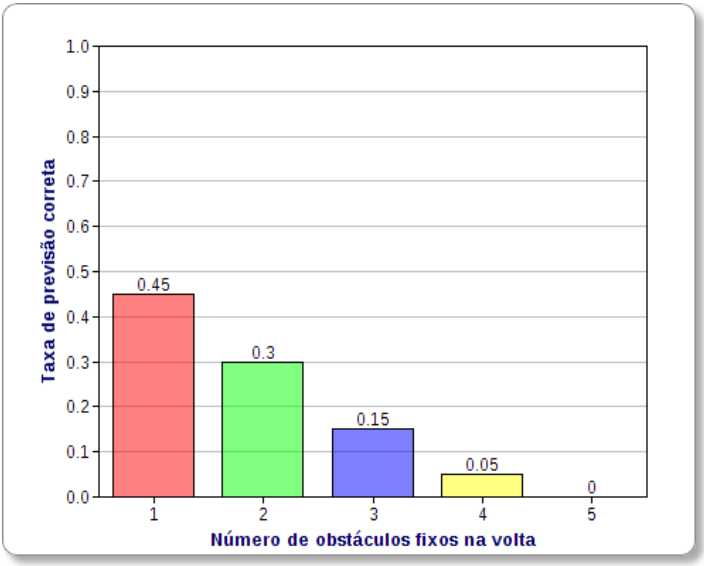

Figura 24: A influência de obstáculos fixos desconhecidos na taxa de previsões corretas no cenário 2, sem compensação

Por fim foram realizados mais 100 testes alterando apenas o número de obstáculos móveis presentes na volta do robô. Em todos os testes, durante a ida até a posição objetivo não foi colocado nenhum obstáculo móvel desconhecido na trajetória do robô. Porém, na volta foi adicionado um obstáculo extra a cada 20 execuções, à semelhança do que aconteceu no caso anterior. A taxa de previsões corretas para cada situação é mostrada na figura 25. Ao contrário do caso anterior, a influência no resultado das previsões não é muito grande, pois em geral o robô perde pouco tempo evitando obstáculos móveis. Os testes realizados consideraram o pior caso em que nenhum obstáculo móvel é detectado na ida, ou seja, não é reservada nenhuma folga para compensar a possível presença desse obstáculo durante a volta. Em outros casos, uma folga é reservada pelo mecanismo, o que reduz ainda mais a influência negativa na qualidade das previsões.

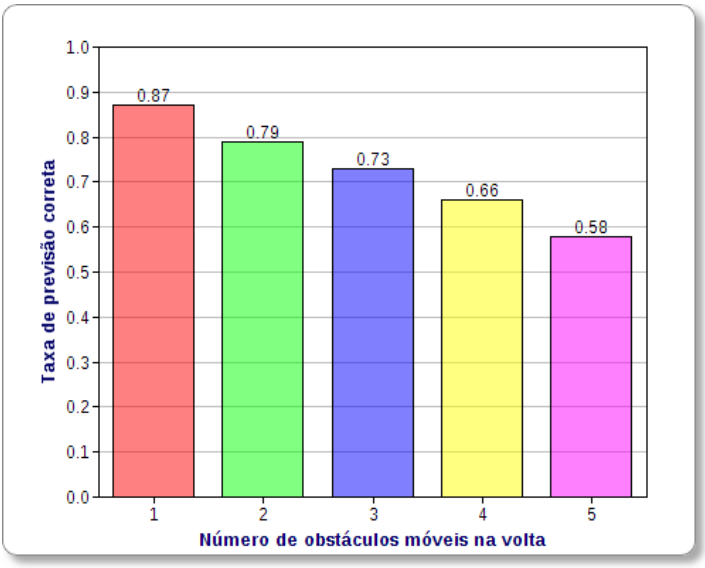

Figura 25: A influência de obstáculos móveis na taxa de previsões corretas no cenário 2, sem compensação

5.4.2.3 Cenário 3

Assim como no cenário 2, neste cenário pode haver mudanças bruscas e repentinas no ambiente. A diferença é que neste caso o módulo de compensação pode ser habilitado automaticamente para compensar tais mudanças. O módulo de compensação é automaticamente habilitado sempre que o mecanismo errar a previsão de cumprimento de *deadline* de duas tarefas consecutivas. O tamanho do histórico de compensação foi definido como 100, mas ao contrário do histórico principal, este tamanho não influencia no resultado da previsão pois apenas alguns poucos registros são utilizados para fazer a compensação. Sempre que o mecanismo errar duas previsões consecutivas, o módulo de compensação é automaticamente habilitado utilizando o primeiro critério de busca com $N = 4$. Assim que o mecanismo de previsão acertar o resultado da previsão, o módulo de compensação é desabilitado novamente até nova ocorrência de dois erros consecutivos. Nos cenários com grande dinamismo no ambiente, ou seja, mudanças frequentes e repentinas no ambiente após o resultado da previsão, não é possível garantir que a taxa de previsões corretas será maior do que 90% pois no momento da previsão

da perda de *deadline* não é possível prever com 100% de certeza se alguma mudança grande no ambiente irá acontecer durante a volta do robô à posição inicial e nem o tempo perdido por causa desta mudança. Porém, o módulo de compensação ajuda a melhorar consideravelmente a qualidade das previsões nestes casos extremos e se as mudanças no ambiente após o resultado da previsão não forem muito frequentes durante a realização de tarefas, é possível obter taxas de previsões corretas na faixa dos 90%.

Neste cenário, todos os testes realizados no cenário 2 foram replicados, mas desta vez com o módulo de compensação habilitado e podendo ser utilizado para compensar as mudanças no ambiente. Foram utilizados os mesmos *deadlines* que foram gerados aleatoriamente nos testes do cenário 2.

Inicialmente foram repetidos os 100 testes realizados no cenário 2 de forma alternada em grupos de 5 tarefas, sendo 5 execuções de tarefas realizadas sem mudança repentina no ambiente seguidas por 5 execuções com mudanças (aumento do número de obstáculos desconhecidos na volta do robô) e assim alternadamente.

5.4.2.3.1 Resultados

Com o uso do módulo de compensação, a taxa de previsões corretas aumentou consideravelmente em comparação com a taxa obtida nas mesmas condições mas com a compensação desligada. A taxa obtida foi de 84%. Neste caso, as mudanças no ambiente foram muito frequentes e a taxa obtida é satisfatória. O gráfico da Figura 26 faz a comparação entre o tempo de volta do robô e a estimativa do mecanismo, nas 100 execuções realizadas no cenário 3. Com o uso da compensação, os tempos obtidos foram consideravelmente mais próximos do que no cenário 2.

Em seguida foram realizados mais 100 testes alterando apenas o número de obstáculos fixos presentes na volta do robô. Em todos os testes, durante a ida até a posição objetivo não foi colocado nenhum obstáculo desconhecido na trajetória do robô. Porém, na volta foi adicionado um obstáculo extra a cada 20 execuções da tarefa. A Figura 27 mostra as taxas de previsões corretas obtidas. É mostrada a taxa para as situações em que foram encontrados de 1 a 5 obstáculos a mais do que o esperado durante a volta do robô. As taxas de previsões corretas aumentaram consideravelmente com o uso do módulo de compensação.

Dando sequência à reprodução dos testes realizados no Cenário 2, foram realizados mais 100 testes alterando apenas o número de obstáculos móveis presentes na volta do robô. Em todas as execuções, durante a ida até a posição objetivo não foi colocado nenhum obstáculo móvel desconhecido

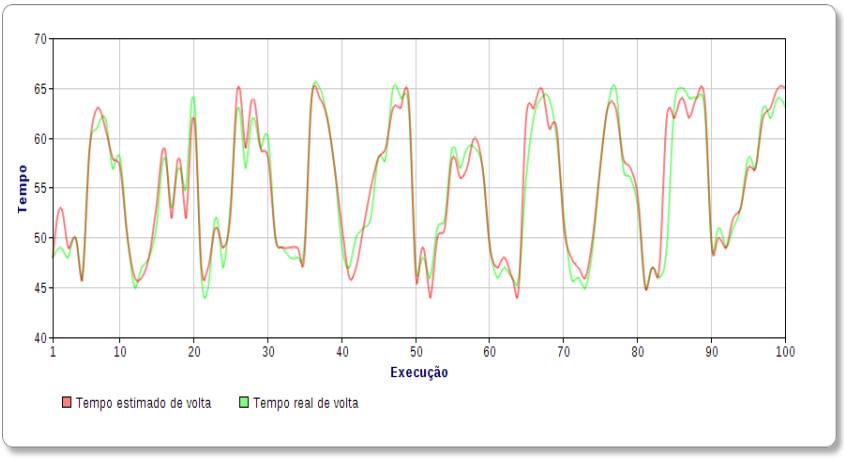

Figura 26: Comparação entre o tempo real de volta do robô e o tempo estimado pelo mecanismo, no cenário 3

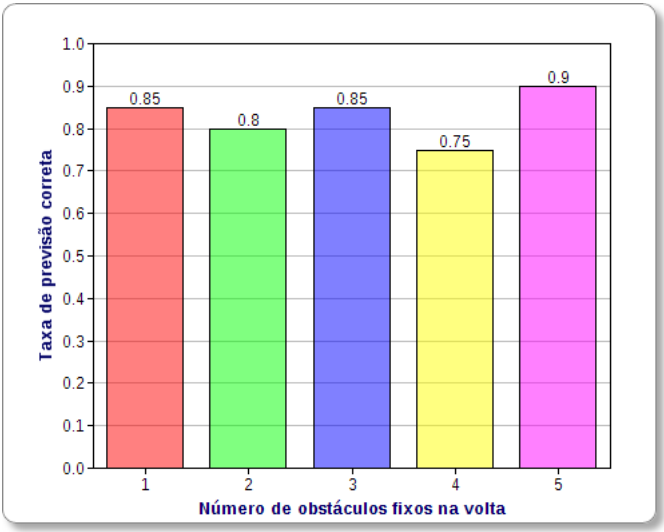

Figura 27: A influência de obstáculos fixos desconhecidos na taxa de previsões corretas no cenário 3, com compensação

na trajetória do robô. Porém, na volta foi adicionado um obstáculo extra a cada 20 execuções. O resultado obtido é mostrado na Figura 28. Apesar do histórico de compensação não armazenar informações sobre a quantidade de obstáculos móveis encontrados pelo robô, o mecanismo consegue compensar de forma indireta a presença de muitos obstáculos móveis inesperados. O histórico de compensação armazena a diferença entre o tempo de volta real do robô e o tempo estimado pelo mecanismo para esta volta. Uma média desta diferença de tempo é utilizada para atualizar a estimativa de volta em caso de previsões incorretas consecutivas. Caso a presença obstáculos móveis provoque o aumento do tempo de volta do robô, este tempo é considerado pelo módulo de compensação na hora de atualizar a estimativa de volta.

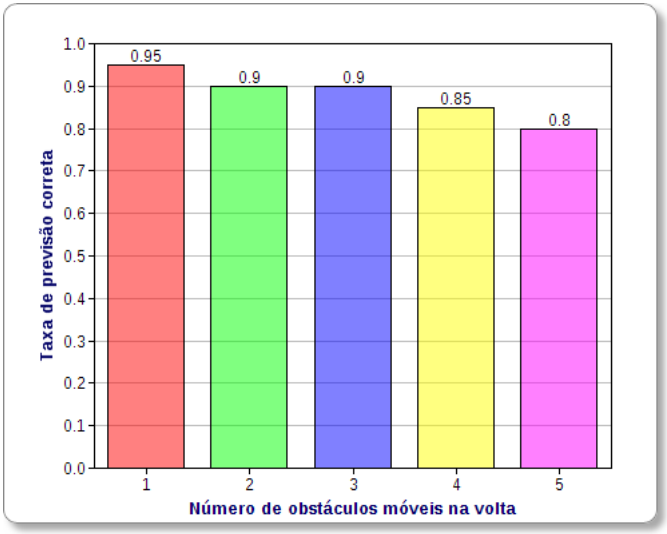

Figura 28: A influência de obstáculos móveis na taxa de previsões corretas no cenário 3, com compensação

5.4.3 Comparação entre simulação e avaliação experimental

Com o intuito de avaliar a diferença entre o desempenho do mecanismo nas simulações e nos testes experimentais, foi feita uma comparação envolvendo 100 execuções no mesmo ambiente, nas mesmas condições e utilizando a mesma sequência de *deadlines*. Primeiro foram feitas 100 execuções experimentais com o robô Pionner 3-DX no ambiente utilizado nos testes an-

teriores (terceiro andar do INE), mas desta vez sem a presença de obstáculos desconhecidos. Em seguida, foi utilizado o simulador MobileSim com o mesmo mapa do ambiente e utilizando os mesmos *deadlines* para as tarefas. Como o simulador do robô é limitado e não suporta ambientes dinâmicos, a comparação só foi possível no cenário onde não se encontram obstáculos desconhecidos durante a realização da tarefa. A Figura 29 mostra as taxas de previsão correta obtidas nos dois casos. O desempenho do mecanismo foi similar nos dois casos.

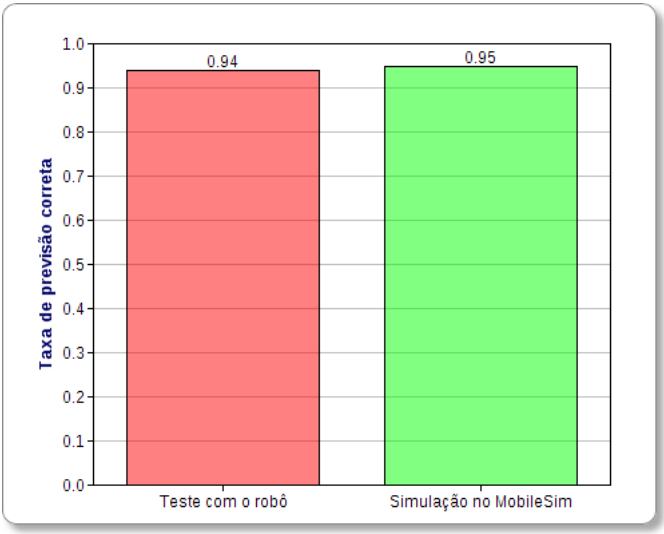

Figura 29: A diferença entre as taxas de previsão correta obtidas nas simulações e nos testes experimentais

Como pode-se observar na Figura 29, os resultados tanto no ambiente real como no ambiente simulado são muito próximos, mostrando a fidelidade do simulador quando reflete um ambiente real.

5.5 VALIDAÇÃO DO MECANISMO

Para validar o mecanismo de previsão de perda de *deadline* proposto é importante compará-lo com outros mecanismos similares. Como não foi encontrado na literatura um mecanismo similar, foi implementado um mecanismo simples e intuitivo e então comparado com o mecanismo proposto nesta dissertação. As próxima seção descreve o mecanismo utilizado para

esta comparação.

5.5.1 Mecanismo para comparação

O mecanismo implementado foi o mecanismo mais simples e intuitivo possível. Um mecanismo que não faz uso de nenhum histórico e faz a previsão através do resultado de uma fórmula simples, conforme mostra a Eq. (5.1). TEP é o tempo de execução parcial, ou seja, o tempo decorrido desde o início da tarefa até o robô atingir a posição objetivo. DL representa o *deadline* da tarefa.

$$TEP \leq DL/2 \tag{5.1}$$

A ideia da equação é a de que o tempo que o robô levou para chegar na posição objetivo será o mesmo tempo da volta para a posição inicial. Logo, a equação compara o tempo de execução parcial com a metade do *deadline*. Se a comparação da equação for verdadeira, o algoritmo assume que haverá tempo suficiente para o robô voltar à posição inicial sem ultrapassar o *deadline* e logo a previsão é de cumprimento de *deadline*. Caso contrário, o algoritmo considera que o *deadline* será perdido.

5.5.2 Comparação entre os mecanismos

A previsão de perda de *deadline* é feita assim que o robô atinge a posição objetivo, à semelhança do que acontece no mecanismo proposto neste trabalho. Para comparar os dois mecanismos, foram considerados 3 cenários distintos. Em cada cenário, os dois mecanismos foram executados utilizando a mesma sequência de *deadlines* para as tarefas, de modo que a comparação fosse a mais justa possível.

5.5.2.1 Cenário 1

Este cenário é o mais simples possível e considera os ambientes simulados e estáticos mostrados na Figura 13, Figura 14 e Figura 15. Foram realizadas 100 simulações para cada mecanismo no MobileSim, exatamente nas mesmas condições. Para o mecanismo de previsão proposto, foi utilizado o mesmo histórico principal de 1000 registros utilizado nas simulações realizadas anteriormente em ambientes simulados. Inicialmente o mecanismo proposto foi executado utilizando como *deadline* o tempo médio total de

execução calculado a partir do histórico principal. Os *deadlines* das tarefas foram armazenados e utilizados nas simulações do mecanismo de comparação.

5.5.2.1.1 Resultados

A Figura 30 mostra as taxas de previsões corretas obtidas nas 3 configurações de ambiente, para os dois mecanismos.

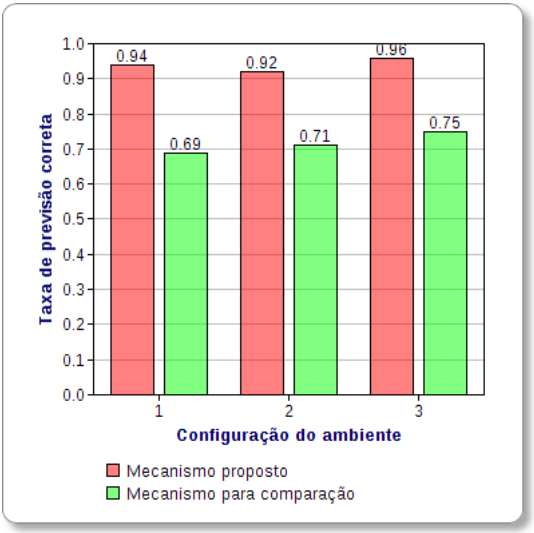

Figura 30: Comparação entre os mecanismos nas simulações em ambientes estáticos

Como visto, as taxas obtidas para o mecanismo proposto foram consideravelmente maiores. Isto se deve ao fato do mecanismo de comparação não fazer uso de um histórico para salvar dados de tarefas anteriores. O sistema de navegação é responsável por enviar comandos ao robô e o instruir a seguir a trajetória planejada. Porém, por melhor que seja o algoritmo utilizado, duas execuções de tarefas similares nunca são idênticas. Sempre existe alguma diferença nas velocidades e orientações escolhidas para o robô, o que leva a uma diferença de tempo na realização de tarefas. O mecanismo de teste considera que o tempo da ida do robô até a posição objetivo é exatamente o mesmo que o tempo da volta. Porém, na prática, isto quase nunca acontece. Nas simulações realizadas observou-se uma diferença de até 8 segundos entre o tempo de ida e o de volta. O tempo de ida se mostrou na maioria das vezes

maior do que o tempo de volta, embora a distância percorrida seja praticamente a mesma. Para melhorar a qualidade das previsões, torna-se necessário armazenar dados de tarefas anteriores em um histórico, de modo a se ter uma média aproximada e mais precisa do tempo estimado de volta do robô.

5.5.2.2 Cenário 2

Neste cenário, o ambiente não sofre alterações repentinas durante a execução da tarefa, ou seja, o ambiente encontrado pelo robô durante a sua ida até a posição objetivo é o mesmo que o encontrado na volta. Isto significa que os mesmos obstáculos desconhecidos encontrados na ida do robô são encontrados na volta até a posição de origem.

Foram realizados 100 testes neste cenário, variando a quantidade de obstáculos desconhecidos fixos, mas mantendo os mesmos obstáculos durante toda a tarefa. De forma contrária, os obstáculos móveis (pessoas) encontrados pelo robô ao longo da sua ida até a posição objetivo, não permaneceram na trajetória do robô durante a sua volta. Ao final de cada tarefa, foi alterado o número e/ou o posicionamento dos obstáculos fixos ao longo da trajetória do robô. De forma aleatória, obstáculos móveis cruzaram a trajetória do robô apenas durante a sua ida.

5.5.2.2.1 Resultados

No final dos 100 testes realizados, foi verificado o resultado das previsões de perda de *deadline* para os dois mecanismos. O resultado pode ser visto na Figura 31. Mesmo sem mudanças repentinas no ambiente, o mecanismo proposto conseguiu uma taxa de previsões corretas 51% maior do que a do mecanismo de comparação.

5.5.2.3 Cenário 3

Neste cenário o ambiente pode sofrer mudanças bruscas e repentinas, ou seja, o número de obstáculos fixos desconhecidos encontrados durante a ida do robô pode variar durante a sua volta até a posição inicial. A mesma variação de número pode ocorrer com os obstáculos móveis.

Foram realizados 100 testes de forma alternada em grupos de 5 tarefas, sendo 5 execuções de tarefas realizadas sem mudança repentina no ambiente seguidas por 5 execuções com mudanças (aumento do número de obstáculos

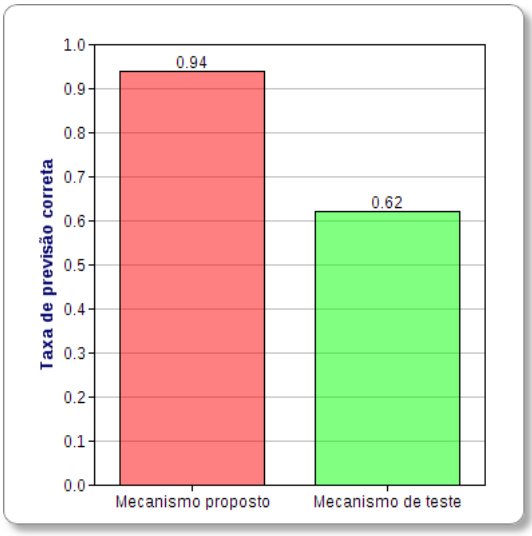

Figura 31: Comparação entre os mecanismos nos testes em ambientes dinâmicos, sem mudança repentina no ambiente

desconhecidos na volta do robô) e assim alternadamente.

5.5.2.3.1 Resultados

Com o módulo de compensação habilitado, o mecanismo proposto conseguiu uma taxa de previsões corretas muito maior do que a do mecanismo de teste. Devido às mudanças repentinas no ambiente, o mecanismo de teste falhou a grande maioria das previsões pois não consegue compensar as mudanças sem o uso de um histórico que armazene dados de execução das tarefas recentes. A Figura 32 mostra as taxas de previsões corretas obtidas para os dois mecanismos.

5.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o hardware utilizado para testar o mecanismo e o software utilizado na sua implementação. Também abordou o módulo de detecção de obstáculos implementado e apresentou as simulações e testes realizados, bem como os resultados obtidos. Os resultados obtidos foram

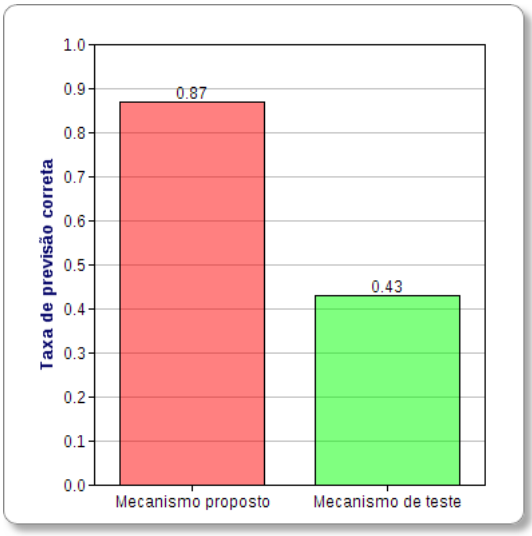

Figura 32: Comparação entre os mecanismos nos testes em ambientes dinâmicos com mudanças frequentes

satisfatórios. Taxas de previsões corretas superiores a 90% foram alcançadas nos ambientes estáticos e ambientes reais com mudanças súbitas não muito frequentes durante a volta do robô à posição inicial da tarefa. Este cenário é o mais provável de acontecer no mundo real, pois as tarefas do robô em geral são relativamente curtas, como por exemplo, ir de uma posição A até uma posição B e voltar à posição inicial, dentro de um depósito.

Para os casos excepcionais em que as mudanças no ambiente são muito frequentes durante a volta do robô, não é possível garantir que a taxa será superior a 90% pois no momento da previsão o mecanismo não tem como antecipar com exatidão uma mudança brusca que ocorra depois do resultado da previsão e nem o tempo que será perdido por causa desta mudança. Se essas mudanças forem muito frequentes e se repetirem durante a realização de muitas tarefas, podem causar algumas previsões incorretas. No entanto, o uso da compensação minimiza de forma substancial a ocorrência de previsões erradas devido a essas mudanças e aumenta a taxa de previsões corretas, que pode ou não atingir os 90%, dependendo da frequência das mudanças e do tempo perdido pelo robô devido a essas mudanças.

O próximo capítulo apresenta alguns mecanismos de previsão já propostos anteriormente em algumas áreas da Computação e os compara alguns com o mecanismo proposto.

6 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns mecanismos e algoritmos de previsão relacionados com a robótica móvel presentes na literatura e os compara com o mecanismo proposto nesta dissertação. A literatura também é abrangente em trabalhos que propõem algoritmos que fazem a previsão de tempos de resposta de tarefas e de desempenho de sistemas. Alguns destes trabalhos também são apresentados neste capítulo.

6.1 MECANISMOS DE PREVISÃO NA ROBÓTICA MÓVEL

Alguns mecanismos já foram propostos para prever a melhor trajetória possível para o robô com uma determinada finalidade, como por exemplo, minimizar o consumo de bateria, navegar com segurança em ambientes pouco conhecidos, interceptar alvos móveis, etc.

Tsubouchi e Arimoto (1994) e Tsubouchi, Hirose e Arimoto (1993) abordam o problema da navegação de um robô em ambientes com múltiplos obstáculos móveis. O mecanismo proposto tenta prever uma trajetória possível para o robô evitar os obstáculos móveis, através de um método de heurística. O movimento de cada obstáculo é previsto assumindo inicialmente que o mesmo possui uma velocidade constante. O algoritmo é iterado frequentemente para acomodar as mudanças reais na velocidade dos obstáculos. Baseado no movimento dos obstáculos, o mecanismo tenta prever a melhor velocidade de navegação para o robô e então armazena o estado do obstáculo e a velocidade do robô prevista. Se o robô for confrontado novamente com uma situação memorizada anteriormente, a velocidade correspondente do robô é então utilizada. Miura, Uozumi e Shirai (1999) propõem um mecanismo aplicável em ambientes com obstáculos estáticos e móveis. O mecanismo tenta prever o futuro movimento dos obstáculos móveis para então planejar uma rota segura e eficiente para o robô. Para a previsão, o mecanismo considera a informação de incerteza. É utilizado um modelo de probabilidade da incerteza e o mecanismo tenta selecionar a trajetória que minimize o tempo para chegada do robô na posição de destino. Shi, Wang e Yang (2010) também apresentam um método para robôs autônomos evitarem obstáculos móveis através da previsão do movimento dos obstáculos e Maeda, Shimakawa e Murakami (1995) apresentam um mecanismo de previsão de rotas seguras em ambientes desconhecidos utilizando a lógica de Fuzzy.

Zhu, Hu e Henschen (2013) apresentam um algoritmo de interceptação de alvos em movimento. O robô pode interceptar um alvo seguindo

muitas trajetórias curtas e de linha reta. No algoritmo, um ponto de intersecção é inicialmente previsto assumindo que o robô e o alvo se movimentam ao longo de trajetórias retas. O algoritmo tenta então planejar um caminho de navegação para o ponto de intersecção previsto. O robô navega ao longo do trajeto planejado, enquanto monitora continuamente o alvo. Quando o robô detecta que o alvo se encontra em nova posição, o mecanismo volta a prever um novo ponto de intersecção e replaneja o caminho de navegação. Este processo é repetido até que o robô intercepte o alvo em movimento.

Mei et al. (2004) apresentam uma abordagem para encontrar trajetórias com eficiência energética para robôs móveis. O mecanismo tenta prever as melhores trajetórias e determinar as velocidades recomendadas para o robô. No mecanismo, a relação entre a velocidade do motor e o seu consumo de energia é modelada através de polinômios. A velocidade do robô é relacionada com a velocidade das suas rodas através de uma transformação linear. O algoritmo faz a comparação do consumo de energia de diferentes rotas em diferentes velocidades e leva em consideração a energia consumida para a aceleração e realização de curvas. Assim, o mecanismo consegue prever a melhor rota que o robô deve seguir e a velocidade recomendada para minimizar o consumo de energia. Um mecanismo similar é apresentado por Jia, Zhou e Chen (2004). Mei et al. (2006) apresentam uma abordagem para a exploração de ambientes desconhecidos por um robô, de forma energeticamente eficiente. O algoritmo proposto determina qual o próximo alvo deve ser visitado pelo robô com base em informações da sua orientação. O mecanismo planeja o caminho entre a posição atual do robô e o próximo alvo de forma eficiente em termos de energia evitando ao máximo que caminhos já conhecidos sejam explorados novamente. No fim, o mecanismo consegue prever a melhor combinação de trajetórias e a ordem correta para o robô percorrer todo o ambiente, consumindo o mínimo de energia possível. Mei et al. (2005) fazem um estudo de caso sobre os principais consumidores de energia em um robô móvel e propõe um método para estimar a duração da bateria e prever o consumo de energia na realização das tarefas de baixo nível do robô. Zhang, Lu e Hu (2009) apresentam um mecanismo para minimizar o consumo de energia de um robô móvel, prevendo a melhor velocidade de locomoção do robô e a frequência do seu processador de bordo simultaneamente. Propõe-se um procedimento geral de solução adequada para as funções arbitrárias de potência do motor e do processador. Sadrpour, Jin e Ulsoy (2012) apresentam um mecanismo para prever a energia necessária para missões de veículos terrestres não tripulados e atualizar a previsão durante a execução da missão através de medições em tempo real do consumo de energia, a velocidade do veículo e outros parâmetros.

Guerrero e Oliver (2012) apresentam um método de alocação de ta-

refas para grupos de robôs que trabalham em equipe. Dado um conjunto de tarefas com um determinado *deadline* total, o mecanismo tenta alocar da melhor forma possível as tarefas a diferentes robôs de um grupo, com o objetivo de minimizar a interferência de um robô na ação dos outros e cumprir o *deadline* total.

Como visto, existem alguns mecanismos e algoritmos de previsão já propostos na área da robótica móvel. A partir dos mecanismos de previsão de duração e de consumo de bateria, é possível prever se um robô cumprirá ou não uma determinada tarefa do ponto de vista da duração da bateria e da energia disponível. Porém, não foi encontrado na literatura nenhum mecanismo genérico como o proposto neste trabalho, para as tarefas de navegação do robô e que considera outras variáveis como a existência de obstáculos no ambiente e um *deadline* a ser cumprido.

6.2 MECANISMOS EM OUTRAS CLASSES DE SISTEMAS

Mecanismos de previsão de tempo de resposta e de desempenho de sistemas são objeto de estudo em várias áreas da computação. A seguir são apresentados alguns desses mecanismos.

Plentz (2008) apresenta três mecanismos de previsão de perda de *deadline* para sistemas tempo real implementados a partir de *threads* distribuídas. O primeiro mecanismo, baseado no conceito de *Milestones*, utiliza informações conhecidas de ativações passadas e define os *Milestones* estaticamente, no nodo origem, antes da *thread* distribuída iniciar sua execução. O segundo mecanismo, baseado na Folga Restante da *thread* distribuída, considera todos os possíveis itinerários que uma *thread* distribuída pode executar, além de informações das execuções anteriores da *thread*. O uso dos possíveis itinerários de uma *thread* ocorre em tempo de execução, através da consulta de uma estrutura de dados, carregada pela *thread* distribuída conforme ela visita os nodos do sistema. O terceiro mecanismo relaciona informações conhecidas de ativações passadas com informações conhecidas em tempo de execução, através do uso de regressão linear. O mecanismo utiliza a composição da fila do servidor de aperiódicas dos nodos que compõem os possíveis itinerários que a *thread* distribuída pode executar. Esta composição se refere aos *deadlines* locais das demais *threads* ativas no sistema. O mecanismo também consulta a estrutura de dados para realizar o cálculo da previsão, além de informações armazenadas no nodo onde a previsão está sendo realizada.

Kirtane e Martin (2006) apresentam o projeto e a implementação de um sistema de gerenciamento de serviços que prevê os tempos de resposta de aplicações web orientadas por transação. O serviço de previsão é iniciado por

um cliente que solicita ao gerente de serviços qual o tempo de resposta de um determinado serviço. O gerente interage com o sistema, calcula o tempo de resposta previsto com base nas condições correntes do sistema e retorna esta informação ao cliente.

Thereska et al. (2006) propõem um mecanismo de auto-previsão de sistemas de armazenamento distribuído. O algoritmo consegue prever o desempenho de uma determinada carga se os seus dados forem movidos do dispositivo A para o B. O sistema gera informações usadas pelos administradores para fins de análise de desempenho.

Tatibana, Montez e Oliveira (2007) apresentam um mecanismo de previsão de cumprimento de *deadline* para sistemas embarcados que executam várias aplicações diferentes. Uma aplicação é composta por um conjunto de serviços. Tarefas com *deadlines* não-críticos requisitam um serviço da aplicação. O algoritmo determina a probabilidade de uma tarefa cumprir seu *deadline*, unindo todos os estados possíveis do sistema em dois: normal ou sobrecarregado. Um histórico armazena o tempo de resposta médio de cada serviço da aplicação. Quando a tarefa finaliza sua execução, seu tempo de resposta é comparado com o tempo de resposta médio associado com o serviço que a tarefa utilizou.

Smith, Foster e Taylor (2004) apresentam um algoritmo de previsão de tempo de execução de uma aplicação paralela. A previsão é baseada nos tempos de execução passados de aplicações paralelas similares. O algoritmo considera aplicações que não possuem restrições temporais.

Na área de sistemas distribuídos de tempo real, Welch, Stoyenko e Marlowe (1995) propõem funções de previsão de tempo de resposta de processos periódicos. O cálculo da previsão é realizado offline e considera o tempo de execução gasto por cada atividade que compõe o processo em recursos hospedados nos nodos do sistema. As funções utilizadas para calcular uma estimativa de tempo de resposta são baseadas no período dos processos. Huh e Welch (2006) propõem técnicas para prever o tempo de resposta de uma aplicação distribuída de tempo real em um nodo. Os métodos consideram o atraso que a aplicação irá sofrer em função de dois tipos de aplicações: aplicações com a mesma prioridade da aplicação em questão e aplicações com prioridades superiores à da aplicação em questão. Os métodos propostos utilizam a técnica *Profiling* para determinar o tempo de execução da aplicação.

A Tabela 4 compara o mecanismo proposto neste trabalho com alguns dos trabalhos descritos anteriormente, apontando as diferenças e características em comum.

Tabela 4: Tabela comparativa entre mecanismos de previsão na Robótica Móvel

	Ambiente	Robô móvel	Previsão	Sensores utilizados	Considera execuções anteriores
Tsubouchi e Arimoto (1994)	Parcialmente conhecido	Terrestre	Trajetória livre dos obstáculos móveis	Variável	Sim
Maeda, Shimakawa e Murakami (1995)	Desconhecido	Terrestre	Trajetória segura	Câmera CCD	Não
Miura, Uozumi e Shirai (1999)	Parcialmente conhecido	Terrestre	Trajetória livre dos obstáculos	Infravermelhos Sonares	Não
Mei et al. (2004)	Parcialmente conhecido	Não definido	Trajetórias com eficiência energética	Variável	Sim
Mei et al. (2006)	Desconhecido	Terrestre	Trajetória pra exploração completa do ambiente	Laser	Não
Zhang, Lu e Hu (2009)	Parcialmente conhecido	Terrestre	melhor velocidade e frequência do processador do robô	Variável	Não
Shi, Wang e Yang (2010)	Parcialmente conhecido	Não definido	Melhor trajetória para desvio de obstáculos	Laser Câmera CCD	Não
Sadrpour, Jin e Ulsoy (2012)	Desconhecido	Terrestre	Energia necessária para as missões	Variável	Não
Guerrero e Oliver (2012)	Totalmente conhecido	Terrestre	Melhor alocação de tarefas a um grupo de robôs	Laser	Não
Zhu, Hu e Henschen (2013)	Parcialmente conhecido	Terrestre	Interceptação de obstáculo móvel	Variável	Não
Mecanismo proposto	Parcialmente conhecido	Terrestre	Perda de *deadline* para tarefas de navegação	Variável	Sim

6.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns mecanismos de previsão presentes na literatura. Constatou-se que na área de robótica móvel ainda não existe um mecanismo similar ao proposto, capaz de prever a perda de *deadline* nas tarefas de navegação do robô. Entretanto, o mecanismo de previsão proposto neste trabalho inspira-se em algumas características dos trabalhos descritos anteriormente. Assim como o método apresentado por Tsubouchi e Arimoto (1994), o funcionamento básico do mecanismo proposto neste trabalho consiste no armazenamento de informações relativas a tarefas já realizadas e na comparação destas com as informações referentes à tarefa em execução. Para armazenar as informações das tarefas já executadas, o mecanismo faz uso de uma estrutura de dados (um histórico) que é consultado em tempo de execução para o cálculo da previsão, assim como os mecanismos apresentados por Plentz (2008) e Tatibana, Montez e Oliveira (2007).

O próximo capítulo desta dissertação apresenta as considerações finais do trabalho e sugere possíveis trabalhos futuros.

7 CONSIDERAÇÕES FINAIS

7.1 CONCLUSÃO

A robótica móvel vem ganhando um papel cada vez mais importante na sociedade moderna. Muitos robôs são desenvolvidos para realizar serviços de utilidade aos seres humanos e auxiliam em tarefas como busca e resgate, assistência doméstica (aspiradores de pó, cortadores de grama, etc), entretenimento (futebol de robôs, robôs que se comportam como animais de estimação, etc), assistência a pessoas com deficiência (cadeiras de rodas robóticas e dispositivos de auxílio ao caminhar) entre outras. A aplicação prática de robôs móveis em diferentes atividades na nossa sociedade, vem demonstrando o quão promissor é o futuro desta linha de pesquisa.

Considerando-se apenas aplicações profissionais (não domésticas), robôs de serviço já formam um mercado de mais de 3,2 bilhões de dólares sendo que 7% do valor refere-se a unidades para aplicações médicas e 75% refere-se à área de defesa. Considerando aplicações domésticas e pessoais (robôs aspiradores, cortadores de grama, de entretenimento, educacionais e de pesquisa), apenas em 2010, foram vendidas 2,2 milhões de unidades num mercado de 540 milhões de dólares. De 2011 a 2014 a previsão de venda de robôs é de 14,4 milhões de unidades apenas para uso pessoal (IFR, 2013).

As restrições temporais presentes nos sistemas de tempo real podem ser aplicadas nas tarefas de navegação dos robôs móveis autônomos nas situações em que prazos são exigidos para o funcionamento correto do sistema. O não cumprimento da tarefa no prazo definido pode acarretar em prejuízos materiais, financeiros ou até humanos em casos específicos. O ambiente em que um robô se locomove e realiza tarefas pode ser dinâmico e nem sempre totalmente conhecido. Isso faz com que nem sempre seja possível prever com antecedência se um robô conseguirá realizar uma tarefa dentro de um prazo estipulado. Neste sentido, torna-se necessário desenvolver mecanismos de previsão de perda de *deadline* para navegação de robôs móveis.

Este trabalho teve como objetivo o desenvolvimento de um mecanismo de previsão de perda de *deadline* para a navegação de robôs móveis autônomos em ambientes estáticos e dinâmicos parcialmente conhecidos. O trabalho apresenta um mecanismo simples porém eficiente, que pode ser utilizado em qualquer robô que utilize uma arquitetura híbrida. Foi projetado um sistema modular em que um módulo utiliza a saída de outros módulos sem depender da implementação dos mesmos. Isso permite que o mecanismo de previsão possa ser utilizado em robôs que utilizam diferentes algoritmos

de localização, navegação e planejamento de trajetória, bem como equipados com diferentes sensores para detecção de obstáculos.

Este trabalho acrescentou uma contribuição à área, pois não foi encontrado um mecanismo semelhante ao desenvolvido na literatura. Apenas mecanismos de previsão de duração da bateria, mecanismos de previsão de rotas energeticamente eficientes e mecanismos de previsão de perda de *deadline* em outras áreas, como sistemas distribuídos. Todos os objetivos definidos inicialmente foram alcançados. Várias simulações e testes foram realizados para validar o mecanismo e o mesmo se mostrou eficiente e conseguiu atingir a taxa de acertos de 90%, como desejado. Para realizar as simulações e testes, foram utilizados os sistemas de navegação, planejamento de trajetória e localização oferecidos pela fabricante do robô Pionner 3-DX. Estes algoritmos são proprietários porém se mostraram rápidos e eficientes. O mecanismo pode ser facilmente adaptado e utilizado em outros robôs e outros ambientes de desenvolvimento porém a sua eficiência, neste caso, depende da qualidade dos algoritmos utilizados para localização, planejamento de trajetória e navegação do robô, bem como da precisão dos sensores utilizados para a detecção de obstáculos.

Um artigo entitulado *Deadline Missing Prediction for Mobile Robots through the Use of Historical Data* contendo os resultados iniciais deste trabalho, foi publicado no ICIRA 2014 (*International Conference on Intelligent Robotics and Applications*). Um artigo com os resultados finais será submetido à revista *Journal of Intelligent & Robotic Systems*.

7.2 TRABALHOS FUTUROS

A partir da pesquisa e desenvolvimento realizados neste trabalho, surgem ideias de possíveis trabalhos futuros. Este projeto pode ser estendido de diversas formas. Dentre as principais medidas que podem ser agregadas destacam-se:

- Tornar os cenários de testes mais complexos, aumentando o dinamismo dos obstáculos móveis.
- Elaborar um mecanismo de previsão de cumprimento de *deadline* para um time de robôs móveis autônomos trabalhando em equipe, em que partes de uma tarefa são alocadas a diferentes robôs e todos precisam cumprir um determinado *deadline* para que o prazo total da tarefa seja cumprido.
- Melhorar o sistema de detecção de obstáculos. O sistema implementado é simples e detecta obstáculos móveis próximos à trajetória do

robô. No entanto, técnicas de previsão de interseção de trajetórias podem ser utilizadas para prever a interseção entre a trajetória do robô e de obstáculos móveis distantes.

- Relaxar os critérios de seleção de tarefas similares para a previsão. O mecanismo considera que as tarefas similares a uma determinada tarefa, são as tarefas em que além de possuírem a mesma posição de origem e destino, os obstáculos conhecidos do ambiente precisam estar nas mesmas posições ou posições muito próximas. Um possível trabalho seria explorar até que ponto se pode relaxar essa exigência com os obstáculos conhecidos, sem comprometer a eficiência do mecanismo.

- Antecipar o cálculo da previsão de perda de *deadline* para o momento mais próximo possível do início da tarefa, sem comprometer a eficiência do mecanismo.

REFERÊNCIAS BIBLIOGRÁFICAS

BORENSTEIN, J.; EVERETT, H.; FENG, L. *Navigating mobile robots: systems and techniques*. [S.l.]: A K Peters, 1996. ISBN 9781568810584.

BRAUNL, T. *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. [S.l.]: Springer, 2008. ISBN 9783540705338.

BROOKS, R. A. *A Robust Layered Control System For a Mobile Robot*. Cambridge, MA, USA, 1985.

CHOSET, H. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. [S.l.]: Prentice Hall of India, 2005. (A Bradford book). ISBN 9780262033275.

DUDEK, G.; JENKIN, M. *Computational Principles of Mobile Robotics*. [S.l.]: Cambridge University Press, 2010. (Computational Principles of Mobile Robotics). ISBN 9780521692120.

FABRIZI, E.; SAFFIOTTI, A. Extracting topology-based maps from gridmaps. In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. [S.l.: s.n.], 2000. v. 3, p. 2972–2978 vol.3. ISSN 1050-4729.

FARINES, J.; FRAGA, J. da S.; OLIVEIRA, R. S. de. *Sistemas de Tempo Real*. IME-USP, SÃ£o Paulo-SP: [s.n.], 2000.

GE, S. *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. [S.l.]: Taylor & Francis, 2006. (Automation and Control Engineering). ISBN 9781420019445.

GUERRERO, J.; OLIVER, G. Multi-robot coalition formation in real-time scenarios. *Robotics and Autonomous Systems*, v. 60, n. 10, p. 1295 – 1307, 2012. ISSN 0921-8890.

HOLLAND, J. *Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine*. [S.l.]: Elsevier Science, 2004. ISBN 9780080477183.

HUH, E.-N.; WELCH, L. R. Adaptive resource management for dynamic distributed real-time applications. *J. Supercomput.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 38, n. 2, p. 127–142, nov. 2006. ISSN 0920-8542.

IFR. *Service Robot Statistics - World Robotics 2013 Service Robots*. 2013. `http://www.ifr.org/service-robots/statistics/`. Acessado em 20/10/2013.

JAHANBIN, M. R.; FALLSIDE, F. Path planning using a wave simulation technique in the configuration space. *GERO, J.S. Artificial intelligence in engineering: robotics and process*, 1988.

JIA, M.; ZHOU, G.; CHEN, Z. An efficient strategy integrating grid and topological information for robot exploration. In: *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*. [S.l.: s.n.], 2004. v. 2, p. 667–672 vol.2.

KIRTANE, S.; MARTIN, J. Application performance prediction in autonomic systems. In: *Proceedings of the 44th annual Southeast regional conference*. New York, NY, USA: ACM, 2006. (ACM-SE 44), p. 566–572. ISBN 1-59593-315-8.

KOPETZ, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. [S.l.]: Springer, 2011. ISBN 9781441982377.

KSHIRSAGAR, S.; SHUKLA, K. A study of motion planning algorithms for mobile robots. *International Journal on Computer Science and Engineering; 2011 Supplement, p34*, 2011.

LAPLANTE, P. *Real-Time Systems Design and Analysis*. [S.l.]: Wiley, 2004. ISBN 9780471228554.

LATOMBE, J.-C. *ROBOT MOTION PLANNING*. [S.l.]: Springer, 1991. (The Springer International Series in Engineering and Computer Science). ISBN 9780792391296.

LEONARD, J.; DURRANT-WHYTE, H. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, v. 7, n. 3, p. 376–382, 1991. ISSN 1042-296X.

LIU, J. *Real-Time Systems*. [S.l.]: Prentice Hall, 2000. ISBN 9780130996510.

MAEDA, M.; SHIMAKAWA, M.; MURAKAMI, S. Predictive fuzzy control of an autonomous mobile robot with forecast learning function. *Fuzzy Sets and Systems*, v. 72, n. 1, p. 51 – 60, 1995. ISSN 0165-0114.

MALL, R. *Real-Time Systems: Theory and Practice*. [S.l.]: Pearson Education, 2009. ISBN 9788131700693.

MATARIC, M. Integration of representation into goal-driven behavior-based robots. *Robotics and Automation, IEEE Transactions on*, v. 8, n. 3, p. 304–312, 1992. ISSN 1042-296X.

MEI, Y. et al. Energy-efficient motion planning for mobile robots. In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 5, p. 4344–4349 Vol.5. ISSN 1050-4729.

MEI, Y. et al. A case study of mobile robot's energy consumption and conservation techniques. In: *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*. [S.l.: s.n.], 2005. p. 492–497.

MEI, Y. et al. Energy-efficient mobile robot exploration. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. [S.l.: s.n.], 2006. p. 505–511. ISSN 1050-4729.

MIURA, J.; UOZUMI, H.; SHIRAI, Y. Mobile robot motion planning considering the motion uncertainty of moving obstacles. In: *Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.: s.n.], 1999. p. 692–697.

MOBILEROBOTS. *ARIA Core Library*. 2013. `http://www.mobilerobots.com/Software/ARIA.aspx`. Acessado em 08/10/2013.

MOBILEROBOTS. *Intelligent robot positioning and navigation software*. 2013. `http://www.mobilerobots.com/Software/NavigationSoftware.aspx`. Acessado em 06/10/2013.

MOBILEROBOTS. *Laser Navigation Package*. 2013. `http://www.mobilerobots.com/Accessories/LaserNavigationPkg.aspx`. Acessado em 07/10/2013.

MOBILEROBOTS. *Mapper3*. 2013. `http://www.mobilerobots.com/Software/Mapper3.aspx`. Acessado em 08/10/2013.

MOBILEROBOTS. *MobileSim Robot Simulator*. 2013. `http://www.mobilerobots.com/Software/MobileSim.aspx`. Acessado em 08/10/2013.

MOBILEROBOTS. *Pioneer P3-DX"*. 2013. `http://www.mobilerobots.com/researchrobots/pioneerp3dx.aspx`. Acessado em 08/10/2013.

NASA. *Mars Science Laboratory*. 2014. `http://www.nasa.gov/mission_pages/msl/index.html`. Acessado em 28/08/2014.

NEGENBORN, R. *Robot localization and kalman filters: On finding your positions in a noisy world*. Dissertação (Mestrado) — Utrecht University, 2003.

NEHMZOW, U. *Mobile Robotics: A Practical Introduction*. [S.l.]: Springer, 2003. (Applied computing). ISBN 9781852337261.

PEDROSA, D. *Mapeamento de Ambientes Estruturados com Extração de Informações Geométricas Através de Dados Sensoriais*. Tese (Doutorado) — UFRN, Natal - RN, 2006.

PIERI, E. D. *Curso de Robótica Móvel*. 2002. `http://www2.ele.ufes.br/~tfbastos/RobMov/ApostilaUFSC.pdf`. Acessado em 09/10/2013.

PLENTZ, P. *Mecanismos de Previsão de Perda de Deadline para Sistemas Baseados em Threads Distribuídas Tempo Real*. Tese (Doutorado) — UFSC, Florianopólis - SC, 2008.

ROMERO, R. *Robôs Móveis Inteligentes: Princípios e Técnicas*. 2011. `http://wiki.icmc.usp.br/images/5/5b/Aula2_SCC5865_RAFR.pdf`. Acessado em 29/08/2014.

SADRPOUR, A.; JIN, J.; ULSOY, A. Mission energy prediction for unmanned ground vehicles. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. [S.l.: s.n.], 2012. p. 2229–2234. ISSN 1050-4729.

SAHA, S. *Introduction to Robotics*. [S.l.]: Tata McGraw-Hill, 2008. ISBN 9780070669000.

SHI, C.; WANG, Y.; YANG, J. A local obstacle avoidance method for mobile robots in partially known environment. *Robot. Auton. Syst.*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, v. 58, n. 5, p. 425–434, maio 2010. ISSN 0921-8890.

SIEGWART, R.; NOURBAKHSH, I.; SCARAMUZZA, D. *Introduction to Autonomous Mobile Robots*. [S.l.]: MIT Press, 2011. ISBN 9780262295093.

SILVA, E.; MENEZES, E. *Metodologia da Pesquisa e Elaboração de Dissertação*. 2005. `https://projetos.inf.ufsc.br/arquivos/Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf`. Acessado em 08/10/2013.

SMITH, W.; FOSTER, I.; TAYLOR, V. Predicting application run times with historical information. *J. Parallel Distrib. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 64, n. 9, p. 1007–1016, set. 2004. ISSN 0743-7315.

TATIBANA, C.; MONTEZ, C.; OLIVEIRA, R. Soft real-time task response time prediction in dynamic embedded systems. In: OBERMAISSER, R. et al. (Ed.). *Software Technologies for Embedded and Ubiquitous Systems*. [S.l.]: Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4761). p. 273–282. ISBN 978-3-540-75663-7.

THERESKA, E. et al. Informed data distribution selection in a self-predicting storage system. In: *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*. [S.l.: s.n.], 2006. p. 187–198.

THRUN, S. et al. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, v. 128, n. 1-2, p. 99–141, 2000.

TSUBOUCHI, T.; ARIMOTO, S. Behavior of a mobile robot navigated by an ldquo;iterated forecast and planning rdquo; scheme in the presence of multiple moving obstacles. In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. [S.l.: s.n.], 1994. p. 2470–2475 vol.3.

TSUBOUCHI, T.; HIROSE, A.; ARIMOTO, S. A navigation scheme with learning for a mobile robot among multiple moving obstacles. In: *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*. [S.l.: s.n.], 1993. v. 3, p. 2234–2240 vol.3.

WELCH, L.; STOYENKO, A.; MARLOWE, T. Response time prediction for distributed processes specified in cart-spec. *Control Engineering Practice*, v. 3, n. 5, p. 651 – 664, 1995. ISSN 0967-0661.

ZHANG, W.; LU, Y.-H.; HU, J. Optimal solutions to a class of power management problems in mobile robots. *Automatica*, v. 45, n. 4, p. 989 – 996, 2009. ISSN 0005-1098.

ZHU, Q.; HU, J.; HENSCHEN, L. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. *Applied Soft Computing*, v. 13, n. 1, p. 539 – 549, 2013. ISSN 1568-4946.